Enabling the cumulant lattice Boltzmann method for complex CFD engineering problems

Von der
Fakultät Architektur, Bauingenieurwesen und Umweltwissenschaften
der Technischen Universität Carolo-Wilhelmina
zu Braunschweig

zur Erlangung des Grades eines
Doktoringenieurs (Dr.-Ing.)
genehmigte

Dissertation

von
Andrea Pasquali
geboren am 17. November 1984
aus Fermo, Italien

Eingereicht am	28. Juni 2016
Disputation am	15. Dezember 2016
Berichterstatter	Prof. Dr. rer. nat. Martin Geier
	Prof. Dr.-Ing. habil. Manfred Krafczyk

2017

No' si volta chi a stella è fisso.
Leonardo da Vinci

To the dreams we have.

Acknowledgment

First and foremost I want to thank Prof. Manfred Krafczyk and Prof. Martin Geier from the Institute for Computational Modeling in Civil Engineering (iRMB) at the Technischen Universität in Braunschweig. It is only thanks to you if I could achieve this goal. Thank you Manfred for giving me the possibility to have the Ph.D. in your institute and to start learning a so interesting CFD method such as the lattice Boltzmann method. Thank you Martin for taking care of this work, for advising me in these years, and for sharing your knowledge with me. I am very proud for being your first Ph.D. student and for sharing the office with you! A special thanks to the whole team: Anna, Martin S., Konstantin, Ehsan, Hesam, Hussein, Jannis, Helmut, Fritz, all the HiWis, and all the colleagues with whom we spent a portion of these years together, Sebastian, Maik, Ying, Wassim, Sonja. You have never denied to help me, and you have made me enjoy the life in the institute!

I want to thank all the people I have left in Italy and who contributed in some way to my professional and personal growth. Thank you Daniele for believing in me after my graduation and for giving to me the possibility to start the "journey" in CFD in your company. Thanks to all my family and friends for supporting me every time and wherever I am. It is always difficult to live far from you but I keep you with me in my heart.

Finally, I want to thank my wife Cristina. We started this adventure together with many expectations and also some difficulties. The way ahead is perhaps still long and not easy. But as Leonardo da Vinci said, "he who is fixed to a star does not change his mind". Let's continue to live our dreams!

Andrea Pasquali
Braunschweig
September 2016

Abstract

Computational Fluid Dynamics (CFD) uses numerical methods to solve problems involving fluid flows, e.g. in automotive engineering, energy, civil engineering, and aerospace. One of these methods is the lattice Boltzmann method (LBM). Beside giving accurate results for a wide range of complex flows, the LBM lends itself to efficient implementations on massively parallel systems such as the general-purpose computing on graphics processing units (GPGPU). This work considers the cumulant LBM, which overcomes some problems of classical LBMs, such as the violation of the Galilean invariance, the spurious coupling of the degrees of freedoms, and the hyper-viscosity. In order to enable the cumulant LBM for complex CFD engineering problems, this work focuses on the discretization of the computational domain and the analysis of turbulent flows in the near-wall region.

The grid generation for LBM deals with several issues such as the second order boundary definition, the free shape grid refinement, and the level wise load balancing for parallel meshes. While discretizing complex geometries, these issues have to be handled simultaneously.

Turbulent flows close to walls are characterized by high velocity gradients in a thin region called the boundary layer. In order to resolve numerically the boundary layer, a grid with a high resolution is required resulting in a high computational cost for the simulation. Empirical functions can be used for modelling the boundary layer, allowing to estimate the quantities at the wall even with coarse grids, and thus increasing the efficiency of the numerical simulation.

In this work, a new grid generator that addresses the LBM discretization issues is implemented. It creates free shape multi-level three-dimensional meshes with second order accurate boundary definition for very complex geometries. It can be applied for generating grids of complex bodies, such as cars, porous media, and urban areas.

Regarding the near-wall region treatment, a new wall function is introduced. It uses local information at the boundary nodes for recovering the quantities at the wall. For generating a proper turbulence for wall bounded flows, a new set of relaxation parameters is introduced, which eliminates the spurious dependence of the error on the bulk viscosity.

In conclusion, this work addresses two important aspects for enabling the cumulant LBM for complex fluid flow problems: grid generation and boundary treatment for turbulent flows.

Zusammenfassung

Computational Fluid Dynamics (CFD) setzt numerische Methoden ein um Strömungsproblemen zu lösen, wie sie z.B. in der Fahrzeugtechnik, der Energietechnik, im Bauingenieruwesen und im Flugzeugbau auftreten. Eine dieser Methoden ist die lattice Boltzmann method (LBM). Die LBM gibt nicht nur akkurate Ergebnisse für verschiedene Strömungproblemen, sondern eignet sich zusätzlisch besonders gut zur Implementierung auf massiv paralleler Hardware wie z.B. general-purpose computing on graphics processing units (GPGPU). Diese Arbeit befasst sich mit der Kumulanten LBM, die einige Probleme der klassichen LBM überkommt, wie z.B. die Verletzung der Galilei-Invarianz, die parasitäre Kopplung der Freiheitsgrade und die Hyper-Viskosität. Um die Kumulanten LBM zu befähigen komplexe Ingenieurprobleme zu lösen befasst sich diese Arbeit mit der Diskretizierung des Rechengebietes und der Analyse der Strömung in der Nähe der Wand.

Die Gittergenerierung für die LBM befasst sich mit verschiedenen Fragestellungen wie der Definition von Randbedingungen zweite Ordnung, der Freistiel-Gitterverfeinerung und der levelweisen Lastbalanzierung. Für komplexe Geometrien müssen diese Fragestellungen gleichzeitig behandelt werden.

Turbulente Strömungen in Wandnähe sind durch starke Gradienten in eine dünnen Schicht charakterisiert die man als Grenzschicht bezeichnet. Um diese Grenzschicht numerische abzubilden sind Gitter mit hoher Auflösung notwendig, was zu teuren Simulationen führt. Empirische Funktionen können benutzt werden um die Werte an der Wand selbst auf gröben Gittern abzuschätzen, was die Effizienz der numerischen Methode erhöht.

In dieser Arbeit wird ein Gittergenerator implementiert, der die Eigenschaften der LBM berücksichtigt. Er generiert freistiel multi-level dreidimensionale Gitter mit Randbedingungen zweite Ordnung für komplexe Geometrien. Er kann benutzt werden um Gitter komplexer Körper wie Autos, porösen Medien und urban Gebieten zu erstellen.

Für die wandnahe Strömung wird eine neue Wandfunktion eingeführt. Sie benutzt lokale Informationen an den Randknoten um die Werte an der Wand zu bestimmen. Um eine geeignet turbulente Strömung zu erzeugen wird ein neuer Satz von Relaxationsparametern eingeführt, der die parasitäre Abhängigkeit des Fehlers von der Volumenviskosität eliminiert.

Zusammengefasst befasst sich diese Arbeit mit zwei wichtigen Aspekten für der Kumulanten LBM im Zusammenhang mit komplexe Ingenieurproblemen: der Gittergenerierung und der Wandbehandlung für turbulente Strömung.

Contents

List of Abbreviations

LLE Linearized Leading Error

AMR Adaptive Mesh Refinement

BAW Bundesanstalt für Wasserbau

BCC Body Centered Cubic

BGK Bhatnagar-Gross-Krook

BTE Boltzmann transport equation

CAD Computer Aided Design

CFD Computational Fluid Dynamics

CLB cascaded lattice Boltzmann

CPU central processing unit

DES Detached Eddy Simulation

DNS Direct Numerical Simulation

dof degrees of freedom

FCM factorized central moments

FPSV-WF frictional partial slip velocity wall function

FSI Fluid Structure Interaction

FVM Finite Volume Method

GPGPU general-purpose computing on graphics processing units

HPC High Performance Computing

HVAC Heating Ventilation and Air-Conditioning

ibb interpolated bounce-back

ILES Implicit Large Eddy Simulation

LBM lattice Boltzmann method

LES Large Eddy Simulation

lhs left hand side

METIS Serial Graph Partitioning and Fill-reducing Matrix Ordering

MRT multiple-relaxation-time

NS Navier-Stokes equations

OpenFOAM Open Source Field Operation and Manipulation

OpenMP Open Multi-Processing

OpenMPI Open Source Message Passing Interface

ParMETIS Parallel Graph Partitioning and Fill-reducing Matrix Ordering

PTB Physikalisch-Technische Bundesanstalt

RANS Reynolds-averaged Navier-Stokes equations

rhs right hand side

sbb simple bounce-back

SGS subgrid-scale model

SI International System of Units

SSM scale similarity models

STL StereoLithography

TBL turbulent boundary layer

URANS Unsteady Reynolds-averaged Navier-Stokes equations

VTK Visualization Toolkit

WALE wall-adapting local eddy-viscosity

List of Symbols

$\vec{u}'$ fluctuating component of the velocity

$C_{\alpha\beta\gamma}$ cumulants

C_f skin-friction coefficient

E turbulence energy

F particle probability distribution function in frequency space

$K_{\alpha\beta\gamma}$ central moments

K von Kármán constant

L reference length

Ma Mach number

Re_τ frictional Reynolds number

Re Reynolds number

U reference velocity

Δt time-step

Δx grid spacing

Λ Ginzburg coefficient

Ω collision term

$\bar{\vec{u}}$ average of the velocity

$\bar{\Delta}$ SGS filter width

$\boldsymbol{\tau}$ deviatoric stress tensor

δ boundary layer thickness

η Kolmogorov scale

κ wave number

$\mathcal{L}$ Laplace transform

μ dynamic viscosity

∇ divergence operator

ν_t eddy viscosity

ν kinematic viscosity

ω molecular collision frequency

∂ partial derivative operator

$\prod$ product operator

ρ fluid density

$\sum$ sum operator

τ_w wall shear stress

τ collision mean free time

θ dimensionless speed of sound squared

ε dissipation of turbulence kinetic energy

$\vec{\Xi}$ microscopic particle velocity in frequency space

$\vec{\xi}$ microscopic particle velocity

$\vec{e}$ stream-wise direction

$\vec{g}$ body force

$\vec{n}$ normal to the wall

$\vec{u}$ fluid velocity

c_{lim} limiter coefficient

c discrete speed

f^* after collision state of the particle probability distribution function

f^{eq} equilibrium state of the particle probability distribution function

f particle probability distribution function

k turbulent kinetic energy

l eddy size

$m_{\alpha\beta\gamma}$ moments

p fluid pressure

q sub-grid-distance

r grid refinement ratio

s slip length

t time

u^+ dimensionless velocity

u_∞ free-stream velocity

u_τ frictional velocity

y^+ dimensionless wall distance

y_w distance to the wall

List of Figures

List of Tables

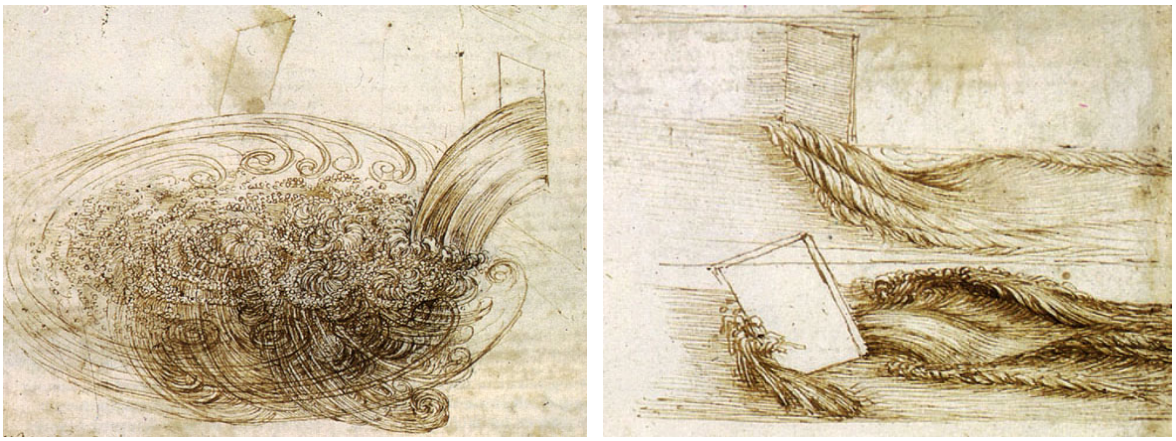
List of Listing

1 | Introduction

I have dream'pt of bloudy turbulence, and this whole night
hath nothing seen but shapes and forms...
Shakespeare, *Troilus and Cressida*

EVERYBODY in their own life has experienced turbulent flows. From children brushing their teeth to a person mixing coffee and sugar with a teaspoon into a cup. From someone laying on the beach and looking at a rough sea to an aircraft pilot flying at more than 800 km/h. Or even more by describing in verses intrinsic sentiments such as love. All of these are examples of chaotic flows and they represent the experience of turbulence.

The first man who made use in his writings of the noun "la turbulenza" (the turbulence) was Leonardo da Vinci early in the 16^{th} century [1]. He was fascinated by the motion of water flows falling down from a hole (Figure 1.1a) and by putting some obstacles into the flow (Figure 1.1b). Although da Vinci was not the first who experienced turbulent flows, he was probably the first one who tried to understand it. Due to the limited knowledge of that time, da Vinci's approach was much more descriptive rather than analytical. In the successive two centuries, the word "turbulence" was mostly a prerogative of writers such as Shakespeare or Wordsworth [1]. The subject of turbulent



(a) (b)

Figure 1.1: Leonardo da Vinci - study of turbulent water flows.

flows and fluid mechanics in general became popular in the 19$^{\text{th}}$ century. Thanks to the works of Claude-Louis Navier, George-Gabriel Stokes, Osborne Reynolds, Joseph Boussinesq, William Thomson, and others [1], nowadays scientists and engineers have access to the equations of motion governing the mechanics of fluids.

The possibility to study and understand the effects of fluid flows is crucial for designing and developing more efficient systems and better products. Indeed, in many engineering fields, the equations of fluid mechanics play a primary role. For example, in automotive engineering they are widely used for external aerodynamics, Heating Ventilation and Air-Conditioning (HVAC), and combustion. They are also used in energy (wind turbine), aerospace, ship, and civil engineering (porous media, pollution, Fluid Structure Interaction). In addition, the equations of fluid mechanics are also required in other fields such as chemistry, medicine, meteorology and astrophysics.

In the recent years, supported by a constant growing in compute capabilities, solving the equations of fluid mechanics was aided by computers, leading to the Computational Fluid Dynamics (CFD). CFD is a branch of fluid mechanics that makes use of numerical methods and algorithms to solve and analyse problems that involve fluid flows. Nowadays, CFD is a standard approach both in industry [2] and academia.

1.1 Fluid mechanics

1.1.1 Navier-Stokes equations

The equations of motion of fluid dynamics solved by Computational Fluid Dynamics are called the Navier-Stokes equations (NS). They describe the mechanics of viscous fluids by a momentum transport equation [3, 4]:

$$\partial_t \vec{u} + (\vec{u} \cdot \nabla)\vec{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \vec{u} + \vec{g}, \tag{1.1}$$

and a continuity equation:

$$\nabla \cdot \vec{u} = 0. \tag{1.2}$$

Eqs. (1.1) and (1.2) are written in incompressible form where $\vec{u} = (u, v, w)$ is the fluid velocity, ρ is the fluid density (constant), p is the fluid pressure, ν is the kinematic viscosity, and $\vec{g} = (g_x, g_y, g_z)$ is an acceleration due to a body force. In contrast to the Euler equations, the NS equations include the effects of viscosity on the flow [3–5]. The terms on the left hand side (lhs) of Eq. (1.1) are the convective time derivative, while those ones on the right hand side (rhs) are the pressure stress and the viscous

stress.

1.1.2 Reynolds number

By defining a similarity parameter Re, it is possible to rewrite Eq. (1.1) in nondimensionalized form [3, 4, 6]:

$$\partial_{t^*}\vec{u}^* + (\vec{u}^* \cdot \nabla^*)\vec{u}^* = -\frac{1}{\rho^*}\nabla^* p^* + \frac{1}{Re}\nabla^{*2}\vec{u}^* + \vec{g}^*, \tag{1.3}$$

where $*$ means the dimensionless form of each variable. Re is called Reynolds number and it is the ratio of the inertial to viscous forces:

$$Re = \frac{\rho U L}{\mu}, \tag{1.4}$$

where U is a reference velocity, L a reference length, and μ the dynamic viscosity ($\nu = \mu/\rho$). The Reynolds number is also an indicative parameter for different flow regimes. If $Re \to 0$ the viscous forces dominate and Eq. (1.3) reduces to a linear diffusion equation (Stokes regime). Thus, for low Re the flow has a smooth and constant motion (laminar flow). On the contrary, if $Re \to \infty$ the inertial forces are dominant. However, Eq. (1.3) does not return to the Euler equations (convection for inviscid fluid) since there will always be a small but not infinitesimally small length scale where the effects of viscosity are important. At high Re the presence of these small but finite scales leads to flow instabilities and chaotic eddies (turbulent flow), whereas no vortices can be generated by the Euler equations [7].

1.1.3 Turbulence energy spectrum

In order to solve numerically Eqs. (1.1) and (1.2) they have to be discretized in space and time and, if the discretizations of the model resolve all the scales of turbulence, the solution of the numerical simulation is known as Direct Numerical Simulation (DNS). This means to resolve the turbulence scales from the largest reference length scale L, associated with the motions containing most of the kinetic energy (inertia), down to the smallest dissipative scales η (Kolmogorov scale):

$$\eta = \left(\frac{\nu^3}{\varepsilon}\right)^{1/4}, \tag{1.5}$$

being a function of the kinematic viscosity ν and the dissipation of turbulence kinetic energy ε [8]. By expressing the scales of turbulence in terms of their wave number $\kappa =$

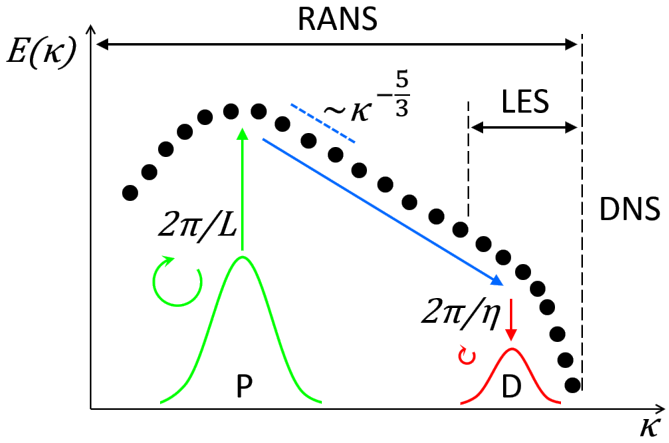


Figure 1.2: Turbulence energy spectrum. The large eddies (with size L) are responsible for the turbulence energy production (P), while the small ones (with size η) for the dissipation (D). The turbulence energy E decays $\propto \kappa^{-5/3}$, being κ the wave number. Direct Numerical Simulation (DNS) resolves all the scales of turbulence, while Large Eddy Simulation (LES) resolves the large scales and model the small ones. The Reynolds-averaged Navier-Stokes equations (RANS) model all the scales.

$2\pi/l$, with l being the eddy size, it is possible to write the energy E associated with the turbulence as function of κ and plot it in spectrum form (Figure 1.2). The turbulence energy has maximum values for large eddy sizes corresponding to the turbulence energy production (P), which is due to the inertial forces. The minimum values are related to small eddy sizes corresponding to the dissipation of turbulence kinetic energy (D), which is due to the viscous effects. Between these two points, E decays with a cascade $\propto \kappa^{-5/3}$ that was first described by Kolmogorov in 1941.

DNS is able to resolve all the turbulence scales providing the right turbulence energy production and dissipation. However, DNS can be applied only for very simple geometries and low Reynolds numbers, because it becomes prohibitive for very high Re. Indeed, in order to resolve down to the smallest turbulence scale η, the grid spacing Δx must obey the condition:

$$\Delta x \leq \eta, \tag{1.6}$$

meaning that the number of grid points N for a three-dimensional DNS must be [9]:

$$N \geq Re^{37/14}. \tag{1.7}$$

For giving an example, a grid for simulating a car driving at 140 km/h ($Re \approx 5 \times 10^6$) should consist of more than 100 quadrillion points (100×10^{15}). For this reason

modelling rather than resolving the small scales of turbulence becomes mandatory.

1.2 Turbulence modelling

1.2.1 RANS

Turbulence modelling started from the works of Osborne Reynolds in 1883 [1, 8, 10, 11]. His idea is to decompose the fluid velocity $\vec{u}$ into two components:

$$\vec{u} = \bar{\vec{u}} + \vec{u}', \tag{1.8}$$

where $\vec{u}'$ is the fluctuating component of the velocity, and $\bar{\vec{u}}$ is the average of the velocity over the time period T:

$$\bar{\vec{u}} = \frac{1}{T}\int_{\frac{t-T}{2}}^{\frac{t+T}{2}} \vec{u}dt, \tag{1.9}$$

with t the time variable. By rewriting the NS momentum equations with the new velocity definition of Eq. (1.8), Reynolds obtained the momentum equation expressed for $\bar{\vec{u}}$ plus a new term depending on $\vec{u}'$:

$$\partial_t\bar{\vec{u}} + (\bar{\vec{u}}\cdot\nabla)\bar{\vec{u}} = -\frac{1}{\rho}\nabla\bar{p} + \nu\nabla^2\bar{\vec{u}} - \nabla\vec{u}'\vec{u}' + \bar{\vec{g}}. \tag{1.10}$$

The six independent components of the new term $\nabla\vec{u}'\vec{u}'$ have the same dimensions of the viscous stresses $\nu\nabla^2\bar{\vec{u}}$ and for this reason they are called Reynolds stresses. The problem is that despite of the six new unknown variables the number of equations is still the same as for the original NS equations. Hence, the Eqs. (1.10) are not close and there is no rigorous way to close them. This is known as the closure problem of the Reynolds-averaged Navier-Stokes equations (RANS). In order to solve the RANS equations, a turbulence model is required assuming a relationship between the mean velocity and the unknown variables. In 1887 Joseph Boussinesq hypothesized such a relationship by introducing an eddy viscosity ν_t:

$$-\nabla\vec{u}'\vec{u}' = \nu_t\nabla^2\bar{\vec{u}}. \tag{1.11}$$

Hence, Eq. (1.10) can be rewritten as:

$$\partial_t\bar{\vec{u}} + (\bar{\vec{u}}\cdot\nabla)\bar{\vec{u}} = -\frac{1}{\rho}\nabla\bar{p} + (\nu+\nu_t)\nabla^2\bar{\vec{u}} + \bar{\vec{g}}, \tag{1.12}$$

where the sum $(\nu + \nu_t)$ represents the effective fluid viscosity.

Over the years, several approaches have been proposed for modelling the eddy viscosity, e.g. the Prandtl's mixing-length concept relying on an algebraic formula (related to the boundary layer), the 1-equation models for the turbulent kinetic energy k, the 2-equations models that consider also the dissipation of turbulence kinetic energy ε [12, 13], and the Reynolds stress equation model that solves one equation for each Reynold stress [11].

By using RANS all the turbulence energy scales are modelled and not resolved (Figure 1.2).

Since the averaging period T for obtaining $\bar{\vec{u}}$ is normally much longer than the largest time scale of the turbulent motion ($T \to \infty$), RANS enables to obtain only steady state solutions and it does not capture the unsteadiness of the flow [8]. For this reason, in order to compute the unsteady "mean" field [8, 14, 15], Unsteady Reynolds-averaged Navier-Stokes equations (URANS) have been introduced.

1.2.2 LES

In order to capture the dynamics of turbulent flows, another approach to turbulence modelling is the Large Eddy Simulation (LES) [16, 17]. LES uses a grid with a resolution that can resolve correctly the large eddies. The small eddies, that have an influence on the large eddies but are smaller than a defined filter width, have to be modelled by a subgrid-scale model (SGS) (Figure 1.2). Therefore, LES performs a low-pass filtering operation in space and not in time (like RANS):

$$\bar{\vec{u}}(x,t) = \int_{-\infty}^{\infty} G_{\bar{\Delta}}(x - x')\vec{u}'(x,t)dx', \tag{1.13}$$

where x and t are the space and time variables, and $G_{\bar{\Delta}}$ is a filter function with width $\bar{\Delta}$. The velocity $\bar{\vec{u}}$ for LES depends on time, and LES is a transient approach. Similarly to RANS, a new stress term is obtained by introducing Eq. (1.13) into Eq. (1.1), which represents the effect of the unresolved velocity components on the resolved ones. Being unknown, this term needs to be modelled with the SGS.

Explicit filtering approaches, such as the Smagorinsky model [18], the dynamic procedure [19], and the scale similarity models (SSM) [20], have been developed in order to deal with the new unknown stress term [17]. The standard Smagorinsky model introduces an eddy viscosity for modelling the influence of the small eddies on the large eddies:

$$\nu_t = (C_s \Delta x)^2 S_{ij}, \tag{1.14}$$

where C_s is the Smagorinsky constant and S_{ij} is the strain stress tensor [18]. The Smagorinsky model is known to over estimate the turbulent viscosity near the solid boundaries. In order to predict correctly the eddy viscosity near the walls, dynamic procedure [19] or wall-adapting local eddy-viscosity (WALE) model [21–23] have to be used.

Since explicit filtering LES models usually apply SGS filter widths proportional to the grid spacing, the new unknown stress term is of the same order of the discretization error. It can therefore be argued that there is no scale separation between the filtering operation and the leading numerical error. Based on this observation the method of implicit filtering was introduced (ILES). It uses stable discretization methods that simply discard all scales smaller than the grid spacing [17].

Interestingly, in the limit of $\bar{\Delta} \to 0$ all the scales are resolved and both LES and ILES turn into a DNS.

LES computations, although being more accurate than RANS, are more expensive because they are naturally transient and requires finer grid resolutions. For this reason, in order to reduce the simulations efforts and still providing satisfactory results, hybrid methods have been introduced in order to use "LES if necessary and RANS if possible". An example is given by the Detached Eddy Simulation (DES) that switches between RANS for modelling the boundary layer and LES for the rest [24–27].

1.2.3 Near-wall treatment

Another important issue of turbulence modelling is the near-wall treatment. Near-wall turbulence is responsible for some important effects such as the friction drag and the wall heat transfer [8]. In particular, in flows with heat transfer, an accurate resolution of the near-wall region is crucial because most of the temperature change occurs across it [28]. The velocity profile in the near-wall region, or boundary layer, is well approximated by a function proposed by Tennekes and Lumley in 1972 ("law of the wall") [29]:

$$u^+ = y^+, \tag{1.15}$$

$$u^+ = \frac{1}{K}\ln(y^+) + B, \tag{1.16}$$

where K is the von Kármán constant and B is a constant ($K \approx 0.41$ and $B \approx 5.0$ for smooth walls). The term y^+ is the dimensionless wall distance:

$$y^+ = \frac{y}{\nu}u_\tau, \tag{1.17}$$

and u^+ is the dimensionless velocity:

$$u^+ = \frac{u_e}{u_\tau}, \tag{1.18}$$

where $u_e = \vec{u} \cdot \vec{e}$ is the velocity in the direction of the stream-wise direction $\vec{e}$. This is defined as:

$$\vec{e} = \frac{\vec{u} - (\vec{u} \cdot \vec{n})\vec{n}}{\|\vec{u} - (\vec{u} \cdot \vec{n})\vec{n}\|}, \tag{1.19}$$

with $\vec{n}$ the normal to the wall. The term u_τ is the frictional velocity:

$$u_\tau = \sqrt{\tau_w / \rho}, \tag{1.20}$$

with τ_w the wall shear stress.

Figure 1.3 shows the velocity profile at the wall by using Eqs. (1.15) and (1.16), and for DNS simulations of turbulent channel flow [30]. The x-axes of the plot is in logarithmic scale and the profile shows four distinct parts [8, 11, 31]. The first one is for $y^+ < 5$ and it is called the viscous sub-layer where the flow regime is laminar and the velocity grows linearly with the distance to the wall (Eq. (1.15) – I). The second part for $30 < y^+ < 200$ is the inertial sub-layer, the flow regime is turbulent, and the velocity grows logarithmically with the wall distance (Eq. (1.16) – II). The region between these two is the transition zone or buffer layer (T), and it is the region where the boundary layer changes from the laminar to the turbulent regime. Finally,

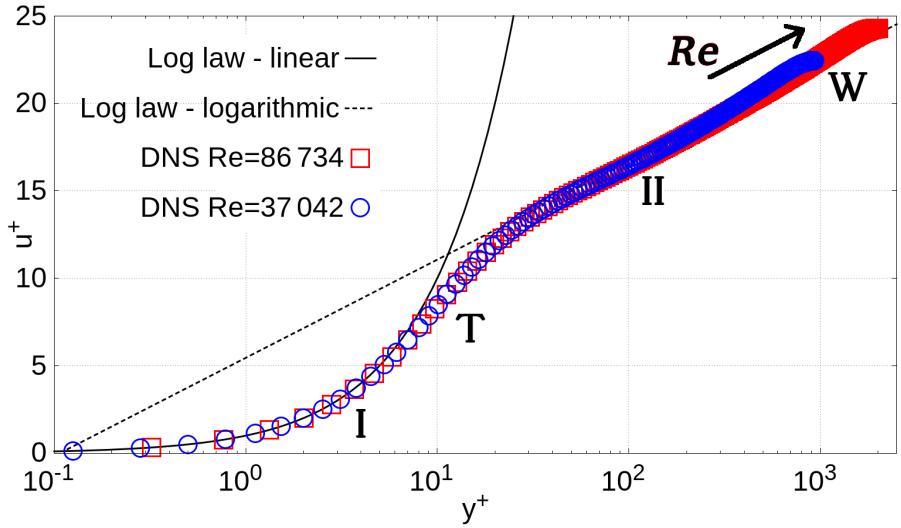


Figure 1.3: Near-wall velocity profile. The velocity profile at the wall by using Eqs. (1.15) and (1.16), and for DNS simulations of turbulent channel flow [30].

the fourth part for $y^+ > 200$ is the turbulent wake region outside the boundary layer (W). Interestingly, the velocity profile in the wake region (W) depends on the Reynolds number, while in the regions I, T, and II it does not. For higher Re the region (W) moves to higher y^+ and u^+, while for lower Re it moves closer to the inertial region (II).

For flows at high Reynolds numbers, the boundary layer is a region of strong velocity gradients because the velocity is zero exactly at the wall and approaches the bulk velocity over a short distance. Due to these strong gradients present in the proximity of the wall, highly refined grids are required in order to resolve the boundary layer [11]. First attempts for dealing with the near-wall turbulence were the introduction of damping functions, leading to low-Reynolds number forms of the standard turbulence models [16]. These methods, although reliable, are very expensive especially for three-dimensional problems [28]. Hence, the modelling of near-wall region flows has become important, initialising the development of the so-called wall functions.By using wall functions the viscous sub-layer does not have to be resolved and very fine mesh is not required.

The first wall function was introduced by Launder and Spalding in 1974 [32]. The idea is to use the law of the wall to compute the wall shear stress τ_w and to adjust the eddy viscosity ν_t for the first grid node accordingly to τ_w. Thus, a wall function simulation normally requires that the first cell outside the walls lies in the logarithmic region ($y^+ > 30$). This approach has been improved by introducing the turbulent kinetic energy k for calculating τ_w [33, 34], either by assuming k as constant in the viscous sub-layer or by taking into account its variations. Other wall functions have been implemented to work properly with smaller y^+ [35, 36], to take the pressure gradients into account [37, 38], and to solve the transport equations in the boundary layer more accurately by adding an extra one-dimensional sub-grid point across each layer of cells [39]. Interestingly, it is also possible to use skin friction boundary conditions as wall functions [40, 41].

1.3 Aim of the work

The development of numerical methods for the CFD can be considered an important topic being as interesting as broad. It covers several fields such as the implementation of physical models, the discretization of the geometries, the management of large data sets, and the coding of efficient and parallel algorithms [42]. Physicists, mathematicians, programmers, and engineers are called to collaborate in order to develop efficient

and scalable methods, allowing the scientific community to satisfy the requests of the current and future challenges.

For carrying out CFD simulations, I opted for the lattice Boltzmann method in its cumulant variant [43] (Chapter 2). The cumulant LBM is a multiple-relaxation-time method, providing accurate results for various fluid flows and allowing efficient implementations on different architectures. In this work, I focused on two different issues of CFD simulations with the cumulant LBM. They are the discretization of the geometries and the analysis of turbulent flows in the near-wall region. The latter was addressed by analysing two aspects, the relaxation parameters, and the near-wall turbulence treatment.

The discretization of the geometries for the LBM requires a Cartesian grid. However, although this can be considered as an advantage [44, 45], the grid generation for complex geometries is not a simple procedure. Second order boundary definition, free shape grid refinement, and level wise load balancing for parallel meshes are only some of the issues that the grid generator has to deal with. I implemented a new grid generator that deals with these challenges creating advanced meshes for very complex geometries (Chapter 3).

Multiple-relaxation-time LBMs deal with the utilization of relaxation parameters for the higher order terms. These parameters can be chosen to improve the accuracy of the results [46, 47]. However, the selection of an optimal set of parameters is difficult since the number of constraints is often larger than the number of parameters, and each set represents some sort of compromise. I addressed this problem by using a new set of relaxation parameters implemented for simulations of turbulent flows at high Reynolds numbers (Chapter 4).

Although being an important aspect of turbulent flows, the near-wall turbulence treatment for the LBM has not been deeply investigated and only few recent studies are available [48]. Wall functions allow to obtain the correct quantities at the wall, e.g. the stresses, even with coarse grids, increasing the efficiency of the numerical simulations. The implementation of a new wall function for the LBM is shown in Chapter 5.

2 | Cumulant LBM

Socrates: All in all, I responded, those who were chained would consider nothing besides the shadows of the artefacts as the unhidden.
Plato, *The allegory of the cave, Republic*

In history the scientific knowledge has grown from the intuitions of scientists that established new breaking-points in the existence of all humankind. it is worth mentioning some eminent people of the past such as Cristoforo Colombo who started an incredible journey after picturing in his mind a rounded Earth, or Galileo Galilei who investigated mathematics and natural philosophy after being attracted by a swinging chandelier, or even Isaac Newton who developed the gravitation theory after being hit by an apple fallen from a tree. Their discoveries were difficult to be assimilated because they bumped into the aversions of the beliefs of those times [49], but once settled they chased away the shadows of the biases and they became the new references for further explorations.

In Computational Fluid Dynamics the lattice Boltzmann method had the same difficulties for becoming popular, but in the last two decades it has been established as a valid method for CFD in both academic and industrial applications [50, 51]. Furthermore, the continuous developing of more sophisticated variants, such as the multiple-relaxation-time (MRT) [52] and the cumulant LBM [43], improved the stability and accuracy of the method, allowing the utilization of the LBM for turbulent flows at high Reynolds numbers.

In this chapter I will briefly introduce the lattice Boltzmann method, the cumulant LBM, and the boundary condition for complex geometries.

2.1 Introduction to the lattice Boltzmann method

The Boltzmann transport equation (BTE), developed by Ludwig Boltzmann in 1872 [53, 54], describes the statistical behaviour of particles by the transport equation:

$$\partial_t f + \vec{\xi}\partial_{\vec{\xi}} f = \Omega, \tag{2.1}$$

where f is the particle probability distribution function, t is the time, $\vec{\xi}$ is the microscopic particle velocity, and Ω is a collision term. The distribution function f can be seen as the probability of a particle to be in a certain state in the momentum space. The lhs of Eq. (2.1) represents the streaming of particles while the rhs is the collision between particles. The collision term Ω is not straightforward to solve. Several attempts have been made to model the collision term, and the first solution obtaining a certain relevance was proposed by Bhatnagar, Gross and Krook (BGK) in 1954 [55]. They assumed that the collision is the process that returns particles to the state of the Maxwellian equilibrium:

$$\Omega = \frac{f^{eq} - f}{\tau}, \tag{2.2}$$

where f^{eq} is the equilibrium state of the particle probability distribution function, and τ is the collision mean free time. Eq. (2.2) is a rate equation with the rate of change (i.e. the molecular collision frequency ω) being related to the collision mean free time as $\omega = 1/\tau$.

To solve numerically Eq. (2.1), it is necessary to discretize it in space, time, and particle velocity. In the lattice Boltzmann method, the particles are allowed only to have discrete velocities. The discrete velocities are chosen such that particles travel a predefined distance in a certain time. When choosing the distances and the length of the time-steps in such a way that particles move from node to node on a Cartesian grid, all interpolations are avoided and the streaming step is exact. Hence, the lattice BTE in three dimensions is written as [56, 57]:

$$f_{ijk(x+ic\Delta t)(y+jc\Delta t)(z+kc\Delta t)(t+\Delta t)} = f_{ijkxyzt} + \Omega_{ijkxyzt}, \tag{2.3}$$

or [43]:

$$f_{ijkxyz(t+\Delta t)} = f^{*}_{ijk(x-ic\Delta t)(y-jc\Delta t)(z-kc\Delta t)t}, \tag{2.4}$$

where ic, jc, and kc are the variables in velocity space $\vec{\xi} = (ic, jc, kc)$, $c = \Delta x/\Delta t$ is the discrete speed and i, j, and $k \in \mathbb{Z}$, and x, y, and z are the variables in space, Δx is the grid spacing, Δt is the time-step, and the symbol $*$ means the post-collision state. Figure 2.1 shows a three-dimensional discretization of the velocity space with 27 discrete speeds, the D3Q27 lattice [58]. The node in the middle (in dark grey) is the source point (velocity zero) from which f streams into the directions of the 26 neighbours (in light grey).The velocity zero is denoted as the resting direction (r). The other directions are east (e) and west (w), north (n) and south (s), and top (t) and bottom (b) for the positive and negative X-, Y-, and Z-direction, respectively. The 20 diagonal directions are combinations of these six. To give an example, the upper

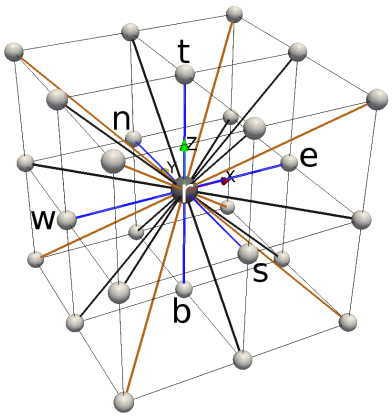


Figure 2.1: The D3Q27 lattice. The point in the middle (in dark grey) is the source point in the resting direction (r). The others (in light grey) are the 26 links ot the directions of the neighbours into the distributions can stream.

corner position in positive Z-, Y- and X-direction is defined as top-north-east (tne). The distance between the source node and any of the neighbours is unitary and thus $i,\, j,\, k \in \{-1, 0, 1\}$.

The lattice Boltzmann method is a numerical method for solving the Navier-Stokes equations and due to the duality of the spatial discretization and the discretization of the velocity space, the LBM fulfils exactly all chosen conservation laws, i.e. conservation of mass, momentum and angular momentum. The hydrodynamic variables such as fluid density ρ, fluid velocity $\vec{u} = (u, v, w)$ and fluid pressure p are the moments $m_{\alpha\beta\gamma} = \sum_{ijk} i^\alpha j^\beta k^\gamma f_{ijk}$ of f and they are computed locally at $xyzt$:

$$\rho = m_{000} = \sum_{ijk} f_{ijk}, \tag{2.5}$$

$$\rho u = m_{100} = \sum_{ijk} i f_{ijk}, \quad \rho v = m_{010} = \sum_{ijk} j f_{ijk}, \quad \rho w = m_{001} = \sum_{ijk} k f_{ijk}, \tag{2.6}$$

$$p = \frac{1}{3} c^2 \rho. \tag{2.7}$$

The kinematic viscosity ν is given through the relaxation rate ω:

$$\nu = \frac{1}{3}\left(\frac{1}{\omega} - \frac{1}{2}\right)\frac{\Delta x^2}{\Delta t}, \tag{2.8}$$

where the term $\frac{1}{3}\left(\frac{1}{\omega} - \frac{1}{2}\right) = \nu_{LB}$ is the kinematic viscosity in lattice units.

Beside its clarity and simplicity, the LBM still presents some difficulties that reside in the collision operator Ω. The solution to Ω given by the single relaxation time

approach of the BGK method became the most popular due to its straightforward implementation. The single relaxation rate is chosen according to the collisions with the highest probability, which are binary collisions. However, due to the high number of discrete velocities, collisions between more than two particles are implicitly included. In the BGK approach multiple particles collisions are given the same probability as binary collisions. This is far from the physical reality and it causes stability and accuracy problems.

In order to improve the method for collisions including more than two particles, multiple relaxation time methods have been introduced. The first multiple-relaxation-time (MRT) method was implemented by d'Humières in 1992 [52, 59]. Although MRT improves the solution of the collision operator, it introduces an additional Galilean invariance violation and hyper-viscosity problems. The first problem is due to the definition of the moments used not being Galilean invariant. The second issue comes from the coupling of the different relaxation rates, because the moments are not statistically independent.Although the hyper-viscosity is formally higher order ($\propto \Delta x^4$), it can not be neglected while solving turbulent flows at high Reynolds numbers because the viscosity defined by ω_1 becomes very small. The numerical diffusion is $\propto \Delta x^2$.

In order to recover the Galilean invariance in the context of multiple-relaxation-time methods, Geier introduced a new moments definition implementing the cascaded lattice Boltzmann (CLB) [60, 61]. The CLB method uses central moments instead of raw moments, it recovers the Galilean invariance, but the variables are still not statistically independent of each other.

Although the hyper-viscosity error in the CLB is of the same order as the general leading order error of the LBM (i.e. it does not reduce the order of convergence), its prefactor can be rather high, such that the results can actually be inferior of those obtained by the BGK [62]. Since the error originates from the spurious coupling of the moments, one possible solution is to use factorized central moments (FCM) [62]. Factorization means decomposition of an object into the product of other objects, decoupling the degrees of freedoms and the moments become statistically independent of each other.

2.2 Theory of cumulants

The factorization proposed in [62] to reduce the hyper-viscosity was theorized with the cumulant LBM [43], which overcomes all the previous problems at once. The cumulant LBM is Galilean invariant, the degrees of freedoms are decoupled, and the

hyper-viscosity is reduced to the same level as in the BGK or lower. The method originates from the question of what is the optimal choice of variables to describe the particle distribution function. In the cumulant LBM, the variables are chosen to be statistically independent of each other. Statistical independence means that the joint probability of the observable variables can be expressed as the product of their individual probabilities. Before writing the joint probability, it is necessary to transform the discrete particle probability distribution function f into a continuous function in frequency space F in order to allow for the Taylor series expansion. The frequency space transformation is done by using the Laplace transform:

$$F(\vec{\Xi}) = \mathcal{L}\{f(\vec{\xi})\}. \tag{2.9}$$

Hence, it is possible to write the joint probability of the observable quantities as the product of their own probabilities (factorization):

$$F = \prod F(C_{\alpha\beta\gamma}). \tag{2.10}$$

Furthermore, in order to perform the Taylor series expansion, the product operator has to be transformed into a sum operator by using the logarithm:

$$\ln(F) = \sum \ln(F(C_{\alpha\beta\gamma})). \tag{2.11}$$

The variables $C_{\alpha\beta\gamma}$ of the function F are defined as countable cumulants:

$$C_{\alpha\beta\gamma} = c^{-\alpha-\beta-\gamma} \frac{\partial^\alpha \partial^\beta \partial^\gamma}{\partial \Xi^\alpha \partial \Upsilon^\beta \partial \mathrm{Z}^\gamma} \ln(F(\Xi, \Upsilon, \mathrm{Z}))\Big|_{\Xi=\Upsilon=\mathrm{Z}=0}, \tag{2.12}$$

where Ξ, Υ and Z are the coordinates of the frequency-velocity-space. Cumulants are statistically independent of each other. Since they are statistically independent, it is admissible that they decay with different rates.

The cumulant LBM calculates cumulants from central moments ($K_{\alpha\beta\gamma}$) and uses them only in the collision [43]. In the collision step each cumulant is relaxed towards its equilibrium with an individual rate $\omega_{\alpha\beta\gamma}$:

$$C^*_{\alpha\beta\gamma} = \omega_{\alpha\beta\gamma} C^{eq}_{\alpha\beta\gamma} + (1 - \omega_{\alpha\beta\gamma}) C_{\alpha\beta\gamma}, \tag{2.13}$$

where $C^*_{\alpha\beta\gamma}$ and $C^{eq}_{\alpha\beta\gamma}$ indicate the post-collision and the equilibrium state of the cumulants, respectively. After collision the central moments are reconstructed and finally

the new populations f^* are recomputed.

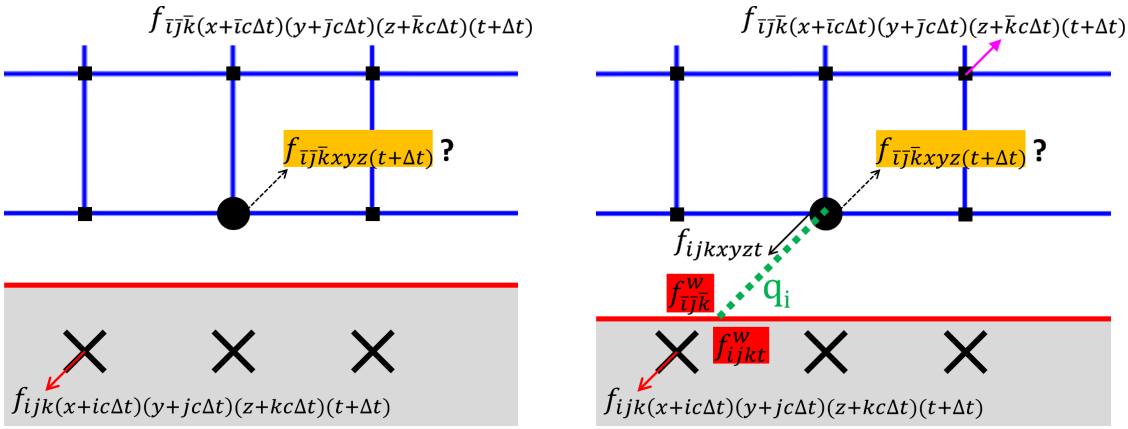
The D3Q27 cumulant LBM collision operator is reported in Appendix A.

2.3 Boundary condition for complex geometries

Due to the utilization of Cartesian grids, the original LBM is not particular suitable for solving flows over complex geometries. For a straight channel flow the walls can be located at the half of the grid spacing and the boundary can be solved by using the simple bounce-back (sbb) approach. But this is not possible for inclined or curved walls. The problem can be overcome by using the cut-cell approach (off-grid) [63], leading to the implementation of interpolated bounce-back (ibb) boundary conditions [64]. Cut-cell means that the real position of the geometry surface is taken into account by defining the distance between the fluid node and the wall. This distance is called sub-grid-distance $q \in \{0 \cdots 1\}$ and it is used for the interpolation.

The sbb boundary condition simply exchanges the unknown distributions coming from the wall by the distributions going into the wall (Figure 2.2a):

$$f_{\bar{i}\bar{j}\bar{k}xyz(t+\Delta t)} = f_{ijk(x+ic\Delta t)(y+jc\Delta t)(z+kc\Delta t)(t+\Delta t)}, \tag{2.14}$$



(a) sbb (b) ibb

Figure 2.2: Boundary conditions for geometries. The fluid nodes are the squares (■), the considered boundary node is the circle (●), and the solid nodes are the crosses (×). The simple bounce-back (sbb) boundary condition (a) exchanges the distributions coming from the wall by the distributions going into the wall, and the wall is located at half grid spacing. The interpolated bounce-back (ibb) boundary condition (b) interpolates between the distributions interpolated at the wall and those ones of the neighbours, and the wall can be located at any ratio of the grid spacing (sub-grid-distance $0 \leq q_{ijk} \leq 1$).

where $\bar{i}\bar{j}\bar{k}$ (coming from the wall) denotes the opposite direction of ijk (going into the wall). By using sbb the wall is a no-slip boundary condition and it is assumed to be half way between the grid nodes. On the contrary, the ibb takes into account the real position of the wall via the sub-grid-distance q using it for the interpolation [43] (Figure 2.2b):

$$f_{\bar{i}\bar{j}\bar{k}xyz(t+\Delta t)} = \frac{1}{q_{ijk}+1} f^{w}_{\bar{i}\bar{j}\bar{k}t} + \frac{q_{ijk}}{q_{ijk}+1} f_{\bar{i}\bar{j}\bar{k}(x+\bar{i}c\Delta t)(y+\bar{j}c\Delta t)(z+\bar{k}c\Delta t)(t+\Delta t)}. \tag{2.15}$$

The interpolation is done between the outward moving distributions at the wall $f^{w}_{\bar{i}\bar{j}\bar{k}t}$ and the distributions of the neighbours in the direction outward the wall $f_{\bar{i}\bar{j}\bar{k}(x+\bar{i}c\Delta t)(y+\bar{j}c\Delta t)(z+\bar{k}c\Delta t)(t+\Delta t)}$. The term $f^{w}_{\bar{i}\bar{j}\bar{k}t}$ considers the velocity at the wall:

$$f^{w}_{\bar{i}\bar{j}\bar{k}t} = f^{w}_{ijkt} - 6w_{ijk}(iu_w + jv_w + kw_w), \tag{2.16}$$

where w_{ijk} are the weight of the corresponding links ($w_{100} = 2/27$, $w_{110} = 1/54$, $w_{111} = 1/216$, and so on by permuting the indices), and $\vec{u}_w = (u_w, v_w, w_w)$ is the velocity at the wall. The term f^{w}_{ijkt} is the inward moving distribution at the wall and it is computed by a linear interpolation between the distributions of the boundary node and those ones entering the wall:

$$f^{w}_{ijkt} = (1 - q_{ijk}) f_{ijkxyzt} + q_{ijk} f_{ijk(x+ic\Delta t)(y+jc\Delta t)(z+kc\Delta t)(t+\Delta t)}. \tag{2.17}$$

By providing $\vec{u}_w = (0,0,0)$ this boundary behaves as a no-slip condition, while by giving $\vec{u}_w \neq (0,0,0)$ the boundary can be used for any velocity conditions, either inflow or slip. Interestingly, by using a sub-grid-distance at half grid spacing like the sbb ($q_{ijk} = 0.5$) and no-slip condition with $\vec{u}_w = (0,0,0)$, the ibb differs from the sbb of Eq (2.14):

$$\begin{aligned} f_{\bar{i}\bar{j}\bar{k}xyz(t+\Delta t)} = {} & \frac{1}{3} f_{ijkxyzt} + \frac{1}{3} f_{ijk(x+ic\Delta t)(y+jc\Delta t)(z+kc\Delta t)(t+\Delta t)} \\ & + \frac{1}{3} f_{\bar{i}\bar{j}\bar{k}(x+\bar{i}c\Delta t)(y+\bar{j}c\Delta t)(z+\bar{k}c\Delta t)(t+\Delta t)}. \end{aligned} \tag{2.18}$$

2.4 Outflow boundary condition

At the outflow of the domain a standard approach is to use an extrapolation boundary condition. The extrapolation exchanges the distributions entering the domain with those ones from the node close to the outlet. Considering for example an outflow

boundary condition in positive X-direction, it is possible to write [43]:

$$f_{\bar{1}jkxyzt} = f_{\bar{1}jk(x-\Delta x)yzt}. \tag{2.19}$$

This approach introduces acoustic reflections that can be reduced by linear interpolation between the distributions at the outlet and the distributions from where the pressure wave came from [43]:

$$f_{\bar{1}jkxyzt} = f_{\bar{1}jk(x-\Delta x)yz(t-\Delta t)}(c\theta^{1/2} - u)\frac{\Delta t}{\Delta x} + f_{\bar{1}jkxyz(t-\Delta t)}\left(1 - (c\theta^{1/2} - u)\frac{\Delta t}{\Delta x}\right), \tag{2.20}$$

where $c\theta^{1/2} = \sqrt{1/3}\Delta x/\Delta t$ is the speed of sound and u is the velocity at the outlet.

3 | Grid generation

Virtue is like a rich stone – best plain set.
Francis Bacon, *Of Beauty*

BEAUTY, such as other virtues, is maybe one of the most fascinating riddles of the human life. Painters, artists, writers, philosophers, theologians, and anyone in the world tried to reach the essence of beauty with their own capabilities and instruments. However, adding artefacts to the originality of beauty, it leads to lose its authenticity. To recover its essence one should strip beauty from any artefacts, thus becoming plain and accessible. This concept could be extended for any thoughts or objects. Numerical methods should respect this principle too, in order to be as plain as possible allowing clear and efficient implementations.

The lattice Boltzmann method has always been considered to allow a straightforward implementation due to its utilization of Cartesian grids [44, 45]. It has the advantage that the discretization of the fluid domain avoids the classic FVM meshing process and the complexity of the geometries is not a limiting factor (e.g. for the prism layers addition). However, this is no longer true for solving very complex problems because the discretization of the geometries, in addition with the use of free shape grid refinement regions, becomes delicate even for the LBM.

I investigated the subject of the LBM grid generation for complex geometries, leading to the development of a new grid generator that complies with the requirements of the cumulant LBM. In this chapter, before explaining the operating principles of the new grid generator, I will introduce the cumulant LBM implementation on GPGPU, highlighting the most important characteristics with regards to grid generation. Finally, I will give some examples and numerical results of complex grids.

3.1 Introduction to the cumulant LBM GPGPU implementation

The cumulant LBM has been successfully implemented on both CPU and GPGPU architectures [65]. It has been used to perform reliable CFD simulations, e.g. for

simulating the dispersion process of ceramic agglomerates on CPU [66] and for solving the external aerodynamics of a generic car on GPGPUs [67]. The implementation of the cumulant LBM on GPGPU has been optimized in order to minimize the amount of data required to define the grid and perform the streaming and collision operations [68].

3.1.1 Data structure

The data structure used in the cumulant LBM consists of a sparse matrix called Esoteric Twist [69], or simply Eso-Twist. For a D3Q27 lattice, the distributions moving in positive directions (e, n, t, ne, te, tn, tne) are stored at the index of the respective source node (r), while the distributions moving in negative directions are stored at the respective neighbour index in the opposite direction (Figure 3.1 – the notation of the distributions for the D3Q27 lattice has been already shown in Figure 2.1). For examples, the distribution with direction (-Z,-Y,-X) (bsw) is stored at the position of the opposite direction (+Z,+Y,+X) (tne). In this way all the distributions are stored only at the nodes in positive directions (the upper octant of the lattice) and only the links to the seven positive neighbours have to be defined for each source point. This number is further reduced to three by the application of a pointer chasing algorithm.

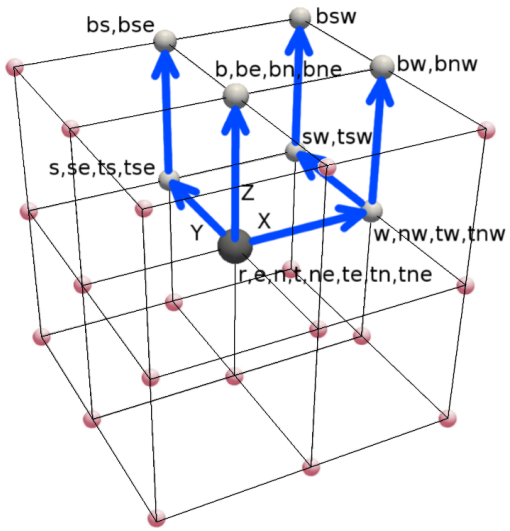


Figure 3.1: Esoteric Twist. The distributions are all stored at the upper octant of the lattice: the distributions going in positive directions are located at the index of the source node (r), while the ones going in negative directions are stored at the index of the opposite neighbour. The entire lattice is further defined by storing only the three neighbours in positive X-, Y-, and Z- directions of the source node. The other distributions are reached by using a pointer chasing algorithm (arrows).

The arrows in Figure 3.1 indicate the pointer chasing paths to reach the neighbours starting form the source point. For example, the neighbour of the source node (r) in direction (+Z,+Y,+X) `nzyx` is reached by taking first the X-neighbour of `r` (`nx`), then the Y-neighbour of `nx` (`nyx`), and finally the Z-neighbour of `nyx` (`nzyx`). Hence, the entire connectivity of the D3Q27 lattice is completely defined by only three neighbours in positive X-, Y- and Z-direction. This means that each node needs the three positive neighbours always defined. This is an important requirement for the grid generator.

3.1.2 Boundary definition

In order to have a second order accurate boundary definition, the cumulant LBM uses a cut-cell approach (off-grid) [63]. Cut-cell means that the lattices close to the boundary are cut by the geometry, and the neighbours of the source node that would be located on the other side of the boundary are missing. Figure 3.2 shows an example for this case. The fraction of the distance between the boundary node and its solid neighbour at which the link is cut is called the sub-grid-distance q. This distance is bounded between 0, where the fluid node resides on the boundary, and 1, where the solid neighbour is on the geometry. The grid generator has to provide this information for any boundary node, taking into account also some particular cases like singularities of the mesh. The calculation of qs is highly demanding, especially for very complex and detailed geometries.

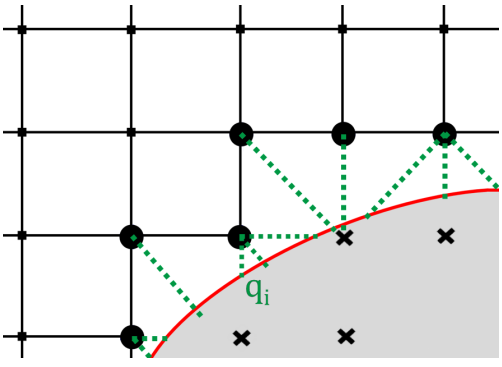


Figure 3.2: Boundary definition. The boundary is defined by using a cut-cell approach. The boundary nodes are the source nodes of the cut lattices (circles) and they own the information of the distance to the boundary in the directions of the missing neighbours ($\times$ symbol), the sub-grid-distance $0 \leq q \leq 1$ (dashed green line).

3.1.3 Grid refinement

For solving complex CFD engineering problems, the resolution of the grid has to adapt to a fine mesh where necessary and to a coarse mesh where possible.By changing the resolution of the grid, the physical quantities such as the speed of sound, the Mach number, and the Reynolds number should remain the same. In order to keep these quantities constant across all grid resolutions, it is a standard procedure to change the time-step lengths Δt proportional to the grid spacing (acoustic scaling). However, by using different Δt, it is necessary to interpolate between grids both in space and time. The cumulant LBM adopts the approach of overlapping grids [70], i.e. only interpolation in space and not in time is required and a quadratic interpolation, necessary for recovering the velocity, is achieved with a compact quadratic interpolation. Moreover, this approach requires staggered grids. Figure 3.3 shows an example of a grid with two different resolutions. There, the positions of coarse and fine nodes do not coincide spatially but they have an offset of ($\Delta x/4$,$\Delta x/4$,$\Delta x/4$) with Δx being the distance between two adjacent coarse nodes. The two grids have an overlapping region for the interpolation in space in both directions, from the coarse to fine grid (CF) and from the fine to coarse grid (FC). On the CF side the eight nodes of the source coarse cell (C) are used to interpolate the eight nodes of the destination fine cell (F). On the FC side the eight nodes of the source fine cell (F) are used to interpolate one single coarse node (C). The interpolation zones (in grey in Figure 3.3) must cover at least five nodes in the finer resolution. A coarse node used as source on the CF side cannot be a destination node on the FC side. In the case of the grid interfaces crossing a

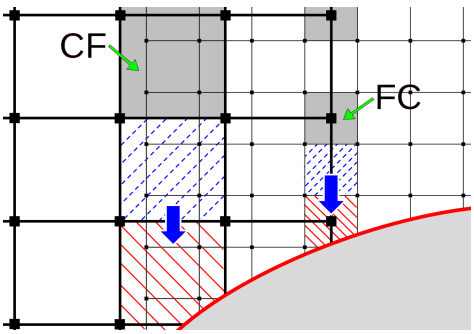


Figure 3.3: Grid refinement interface. Grid with two resolutions, the coarse one (on the left, thick lines and thick points) and the fine one (on the right, thin lines and thin points). The grids are staggered in space and they overlap. The overlapping zone is used for the spatial interpolation in both directions (grey areas), from the coarse to the fine grid (CF) and from the fine to the coarse grid (FC). Cells cut by a boundary are invalid for the interpolation (line filled areas) and an offset to valid cells must be provided (dashed line filled areas).

boundary, due to the cut lattices, the cells are not valid for the interpolation. An offset to the first valid source cell is required. Figure 3.3 shows also this particular case for both offsets on CF and FC sides. Due to the symmetry of the interpolation zones, the technique is independent of the orientation of the interface and does not require any special treatments for corners. Moreover, the use of an overlapping region occupies less memory on GPGPU than an interpolation in time without overlapping region [65]. The construction of this type of mesh refinement results to be an important constrain for the grid generator, especially where the grid interface crosses a boundary.

3.1.4 Multi-GPGPUs

In order to solve large scale simulations, the parallelization of the LBM kernel is essential. The cumulant LBM implementation allows parallel computations on multiple GPGPUs, by transferring the nodal distribution function f from GPGPU to GPGPU [68]. In order to perform this operation, the kernel should know which grid node to send and which grid node to receive at the interface. Figure 3.4 shows the communication between two GPGPUs. There, the first GPGPU has index 0 (bottom) while the second one has index 1 (top). The domains that are allocated on each GPGPU duplicate the "send" nodes (S) of the other GPGPU with "receive" nodes (R). The communication between GPGPUs is ordered, first in X-, then in Y- and finally in Z-direction. The grid generator should create the extra nodes only for parallel cases and store their indices for the communication.

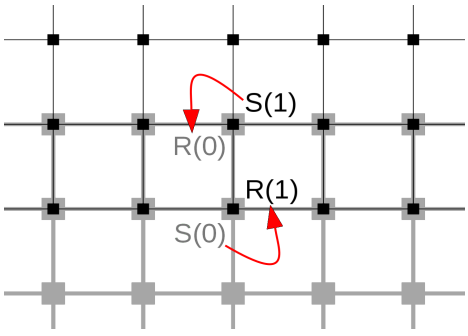
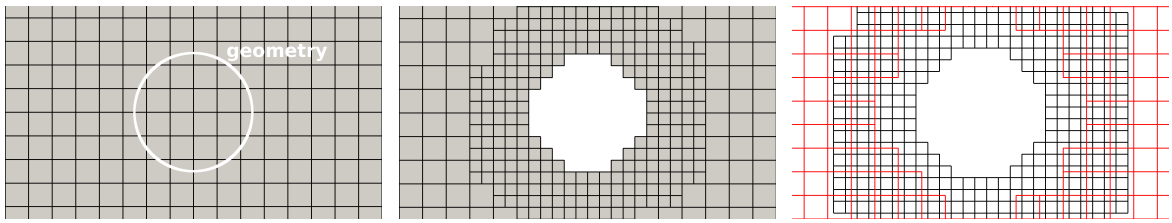


Figure 3.4: Multi-GPGPUs implementation. Communication between two GPGPUs, a first one with index 0 (bottom, in thick grey) and a second one with index 1 (top, in thin black). The domains allocated on each GPGPU have duplicated nodes, "receive" (R), which receive the information coming from the "send" nodes (S) from the other GPGPU.



(a) (b) (c)

Figure 3.5: `LBMHexMesh`. The cumulant LBM mesh generation starts from a background mesh and a geometry (a). The two steps are the refinement stage (b) and the addition of an overlap region (c).

3.2 LBMHexMesh

`LBMHexMesh` is a parallel LBM grid generator that creates free shape multi-level three-dimensional grids with second order accurate boundary definition [71]. Its purpose is to provide grids for the cumulant LBM, precisely for its implementation on GPGPU. `LBMHexMesh` is based on the open-source OpenFOAM grid generator `snappyHexMesh` (release version 2.2.0) [72]. The grid generator `snappyHexMesh` has a powerful algorithm for the grid refinement and its source code, being open, can be modified for obtaining LBM grids. Starting from a background mesh and one or more geometries (Figure 3.5a), `LBMHexMesh` first refines the grid (Figure 3.5b) and then adds the overlap grids for the interpolation (Figure 3.5c).

3.2.1 Background mesh

The background mesh is a Cartesian mesh covering the entire simulation domain. It is generated by using the standard OpenFOAM `blockMesh` application. Figure

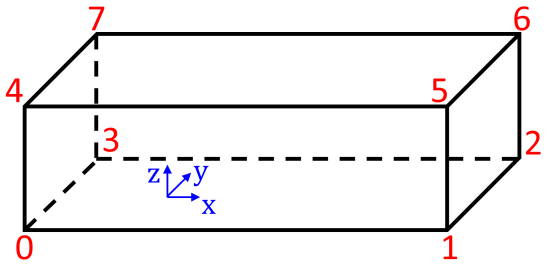


Figure 3.6: Background mesh. The background mesh is built with a single box by defining the eight vertices, the number of cells in X-, Y-, and Z-direction, and the six boundary faces.

```
convertToMeters 1; //scale factor
vertices //eight vertices of the bounding box
(
    (-15.0 -9.0 -0.3) //node index 0
    (45.0 -9.0 -0.3) //node index 1
    (45.0 9.0 -0.3) //node index 2
    (-15.0 9.0 -0.3) //node index 3
    (-15.0 -9.0 14.7) //node index 4
    (45.0 -9.0 14.7) //node index 5
    (45.0 9.0 14.7) //node index 6
    (-15.0 9.0 14.7) //node index 7 //Length = 60m, Width = 18m, Height = 15m
);
blocks //definition of the bounding box and number of divisions
(
    hex (0 1 2 3 4 5 6 7) (200 60 50) simpleGrading (1 1 1)
    //cell size = 0.3m --> 200,60,50 divisions in X, Y, Z
);
edges ( );
boundary //six external faces
(
    inlet  { type wall; faces ( (0 4 7 3) ); } //-X boundary
    outlet { type wall; faces ( (2 6 5 1) ); } //+X boundary
    front  { type wall; faces ( (1 5 4 0) ); } //-Y boundary
    back   { type wall; faces ( (3 7 6 2) ); } //+Y boundary
    bottom { type wall; faces ( (0 3 2 1) ); } //-Z boundary
    top    { type wall; faces ( (4 5 6 7) ); } //+Z boundary
);
```

Listing 3.1: `blockMeshDict` file example.

3.6 gives an example and Listing 3.1 shows the dictionary[1] `blockMeshDict` used to define a background mesh in OpenFOAM. The information necessary for building the background mesh is: i) the eight vertices of the bounding box, ii) the number of divisions in X-, Y-, and Z-direction (according to the cell size wanted), and iii) the six external boundary faces. The `blockMesh` grid generator allows for arbitrary cell aspect ratio. For the LBM, grids with unity cell aspect ratio are required. This has to be taken into account by choosing the correct number of divisions of the domain. `LBMHexMesh` accepts only background meshes with cell aspect ratio equal to one. The external faces can be merged together, e.g. combining the `front`, `back`, `bottom`, and `top` faces into a single entity `sides`. The `type` of the face is not used in `LBMHexMesh` because the boundaries are defined by using a new dictionary file introduced for setting the LBM case (Listing 3.8).

3.2.2 Declaration of the meshing stages

`LBMHexMesh` uses the dictionary `LBMHexMeshDict` for setting all the parameters for the mesh generation. The execution of the two stages of the grid generation, the

[1] In the nomenclature of OpenFOAM a dictionary is a text-file containing definitions of variables and parameters.

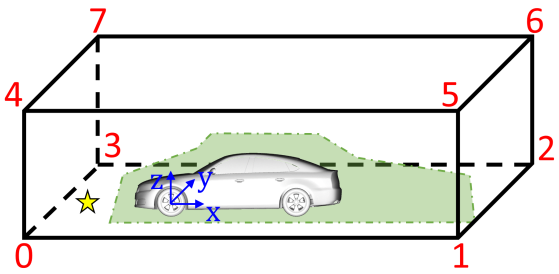


Figure 3.7: Geometry declaration. The geometry (car) and the wake refinement region (green area dashed line surrounding the car) are declared in the dictionary file for the grid generation. The ⋆ symbol is the "keep point" defining the region to be meshed.

stage of refining the grid (`refMesh`) and the stage of generating the overlap grids (`LBMMesh`), is declared in the first part of the file (Listing 3.2). Usually, starting from the background mesh both the steps have to be performed (both `true`).

3.2.3 Declaration of the geometry

The `LBMHexMeshDict` file continues with the declaration of the geometries used for the mesh generation. In the example of Figure 3.7 and Listing 3.3, StereoLithography (STL) [73] geometries are used and they can be of two types: surface geometries and refinement region geometries. The surface geometries are used as boundaries (e.g. the car), and they have to be of type `triSurfaceMesh`. The refinement region geometries are used as refinement regions (e.g. the wake – green area dashed line surrounding the

```
refMesh true; //stage 1
LBMMesh true; //stage 2
```

Listing 3.2: `LBMHexMeshDict` file example: part 1/4 – declaration of the meshing stages.

```
geometry
{
    car.stl //using STL as boundary
    {
        type triSurfaceMesh;
        name car; // same as fine name
        patchInfo { type wall; }
    }
    wake.stl //using STL as refinement region
    {
        type closedTriSurfaceMesh;
        name wake; // same as file name
    }
}
```

Listing 3.3: `LBMHexMeshDict` file example: part 2/4 – declaration of the geometry.

car), and they have to be of type `closedTrisurfaceMesh`. If the STL geometries are composed of different regions, it is possible to specify different refinement levels for each part. For `LBMHexMesh`, the name of each geometry must coincide with the name of the STL file. Usually, STL files are necessary for very complex geometries such as cars, planes, or turbines. For discretizing simpler geometries such as spheres, boxes, or cylinders, it is possible to define the geometry by using the `searchableSurfaces` OpenFOAM library.

3.2.4 Refinement stage

Similarly to the first stage of `snappyHexMesh`, the first step of `LBMHexMesh` performs the grid refinement and the separation of the fluid from the solid cells.

The Listing 3.4 shows the refinement stage parameters.

The first entry is the cells size of the background mesh (`cellSize0`) used for verifying the correct generation of the Cartesian background mesh. Setting a different cell size from the one specified through the declaration of the background mesh results in a fatal error.

The Listing continues with four parts: i) refinement levels declaration for the edges (`features`), ii) refinement levels declaration for the surfaces (`refinementSurfaces`), iii) refinement levels declaration for the regions (`refinementRegions`), and iv) other entries.

The `features`[2] entry allows to use edges from an external file to perform extra refinement iterations specific for the cells intersecting these features. They can be extracted from the geometry with the OpenFOAM command `surfaceFeatureExtract`.

The `refinementSurfaces` entry allows to set the refinement level for each geometry identified by its name as declared in the `geometry` section. The refinement operations are iterative, and they refine the mesh starting from the background mesh level (0) to the desired refinement level ($N \in \mathbb{N}$). The relation between the cell size Δx and the refinement level N is:

$$\Delta x_N = \frac{\Delta x_0}{2^N}. \tag{3.1}$$

In OpenFOAM the cells selected for surface refinement are identified by doing a cell-surface intersection query. If a cell intersects a surface with a resolution different from the current cell resolution, the cell is marked for refinement. The query is done for all the cells in the domain, independent of the position. `LBMHexMesh` implements a

[2] In OpenFOAM geometrical edges are called `features`.

```
refMeshControls
{
    cellSize0 0.3;
    features
    (
        {
            file "car.eMesh"; //external features file
            level 6;
        }
    );
    refinementSurfaces
    {
        car // geometry used as boundary
        {
            level (6 6); //min and max ref. levels
            regions
            {
                underbody { level (7 7); }
                wheelsF { level (7 7); }
                wheelsR { level (7 7); }
            }
        }
        wake // geometry used as ref. region (zone)
        {
            level (0 0); faceZone wake; cellZone wake; cellZoneInside inside;
        }
    }
    refinementRegions
    {
        wake { mode inside; levels ( (1e15 5) ); }
        car { mode distance; levels ( (0.050 6) (0.150 5) ); }
    }
    resolveFeatureAngle 45; //only if min and max ref. levels differ
    allowFreeStandingZoneFaces false; //"false" to use STL as ref. region
    locationInMesh ( (-10 0 0) ); //keep point for fluid cells
    nBufferLayers 5;
    createCoarse 1;
    createFine 3;
    checkRefMeshRefinementStage false;
    checkNeedleCells close;
    loadBalance true;
}
```

Listing 3.4: `LBMHexMeshDict` file example: part 3/4 – refinement stage.

new surface treatment that confines the query operation only to those cells that are inside the bounding box of each geometry. This simple implementation showed to speed up the cell-surface intersection query, especially if many basic surfaces are used with the `searchableSurfaces` library. For each surface it is possible to specify two refinement levels, a minimum and a maximum value. OpenFOAM uses the minimum as default resolution. The maximum resolution is used if the angle of the geometrical features exceeds the angle specified by the `resolveFeatureAngle` entry. This angle is not considered if the minimum and maximum values for the refinement level are equal. If the geometries are composed of multiple regions, each region can be declared with a specific refinement level as for example the `regions` of the geometry `car` in Listing

3.4. The geometries used as refinement regions have to be defined as zones, e.g. the geometry `wake` in Listing 3.4.

In the `refinementRegions` entry it is possible to set the refinement level for the regions, e.g. for the `wake`. In order to use STL files as refinement regions, e.g. the `wake`, the entry `allowFreeStandingZoneFaces` has to be set to `false`. In the example of Listing 3.4, by using the refinement mode `inside`, all the cells inside the region `wake` are refined to level `5`. The surface geometries can also be used as refinement regions, for example through the `distance` option. Considering the example of Listing 3.4 for the geometry `car`, all the cells up to 50 mm and 150 mm away from the geometry surface are refined to level `6` and `5`, respectively. When using refinement regions in `distance` mode, `LBMHexMesh` checks if the distances are set properly to build the correct width of the interface zone, and fixes them automatically if not.

At the end of the refinement stage, `LBMHexMesh` separates the fluid cells from the solid cells, and the latter are removed. This is made possible by the "keep point" declared in the `locationInMesh` entry. This point must be inside the domain but outside the geometry (the $\star$ symbol in Figure 3.7). Only the cells having the centre inside the region of the "keep point" are kept.

One important OpenFOAM entry modified from the standard version is the `nCellsBetweenLevels`. It specifies the minimum width of a refinement level in number of cells. In `LBMHexMesh` this attribute has been replaced by the attributes `nBufferLayers`, `createCoarse`, and `createFine`. The `createCoarse` declares the width at the fine to coarse (FC) interface side, `createFine` is the width at the coarse to fine (CF) interface side, and `nBufferLayers` is the number of cells in between. To obtain the overlap region required by the cumulant LBM, the `createCoarse` and the `createFine` must be at least `1` and `3`, respectively. The `nBufferLayers` entry must be at least `2` to avoid that the overlap regions intersect.

The last three entries of Listing 3.4, `checkRefMeshRefinementStage`, `checkNeedleCells`, and `loadBalance`, are new and specific for the cumulant LBM mesh. They will be explained in more detail below, the first two in this subsection and the third one in the parallel mesh generation subsection.

Checking the overlap width

The interface region between two adjacent levels can have any shape. This is an advantage because it allows to refine very complex geometries efficiently. However, the required interface width is not necessarily satisfied. This happens in particular at concave

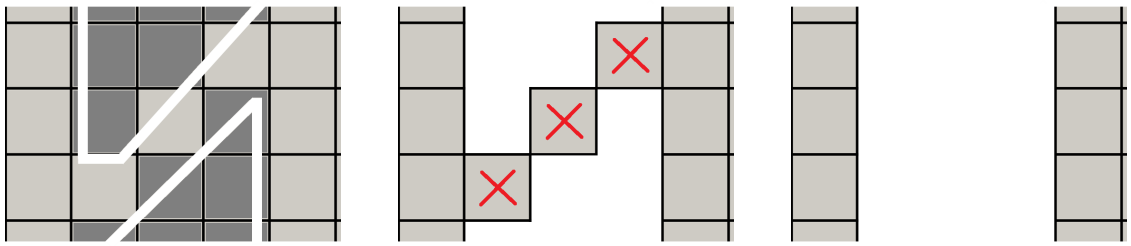
(a) (b)

Figure 3.8: Checking the overlap width. Checking of the width of the interface for the cumulant LBM mesh. If a cell does not have space to build the interface, it is marked (a) and split (b). The white line represents the geometry and the dark-grey cells are solid.

corners of the grid interface. For this reason `LBMHexMesh` implements a new algorithm for checking the width of the overlap region. Figure 3.8a shows a mesh where the changing of the resolution from fine to coarse and from coarse to fine occupies only one coarse cell in vertical direction (marked with the $\times$ symbol). There, the space occupied by the coarse cell is not sufficient for building the required overlapping interface and thus the cell is marked and split (Figure 3.8b). The entry `checkRefMeshRefinementStage = true` specifies that the algorithm is executed in each iteration of the refinement stage. If set to `false` the algorithm is executed only at the end of the refinement stage. By using the latter case the computational time for creating the grid is reduced.

Checking for needle cells

If the geometry has very narrow features but the resolution chosen is not sufficient to resolve them, the grid generator might create some needle cells, like in Figure 3.9a. Needle cells are fluid nodes that do not have fluid neighbours on two or more opposing sides (Figure 3.9b). These cells represent a critical problem for the final LBM mesh,



(a) (b) (c)

Figure 3.9: Checking for needle cells. Due to the narrow geometry (white line) and the poor grid resolution, the fluid cells form a needle channel (a). This is a problem for the cumulant LBM kernel and these cells are selected (b) and removed (c). The dark-grey cells are solid.

```
LBMMeshControls
{
    createSolid auto; //or "all"
    renumberMesh true;
    assignBoundary blockMeshly; //or "geometrically"
    exportWallNormals false;
    setBcFixAsGeom true;
    pressureCheck true;
}
```

Listing 3.5: `LBMHexMeshDict` file example: part 4/4 – overlapping grids stage.

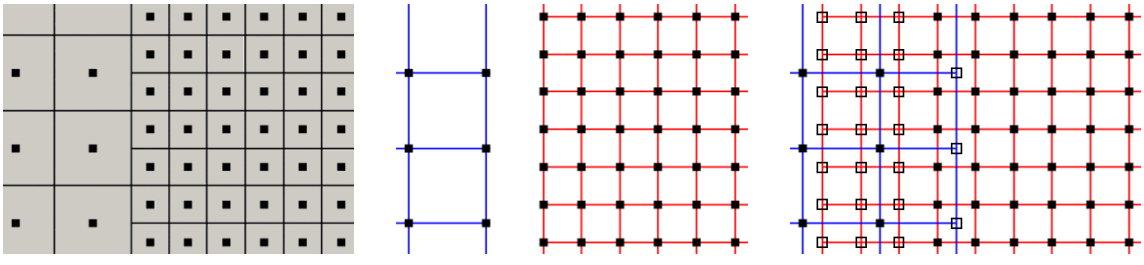
leading to instability of the simulation. `LBMHexMesh` implements an algorithm that, at the end of the refinement stage, when the solid cells are already removed, checks whether needle cells are still present. The entry `checkNeedleCells` can be set only to `close`. The algorithm looks for needle cells, marks them (× symbol in Figure 3.9b), and removes them (Figure 3.9c).

3.2.5 Overlapping grids stage

The second stage of the grid generation is completely new and specific for the cumulant LBM. Listing 3.5 shows the input parameters for this stage.

The first operation is to generate grids of staggered nodes. This is done by replacing the centres of the aligned cells coming from the refinement stage (■ symbol in Figure 3.10a) with the lattice Boltzmann grid nodes (Figure 3.10b).

Each lattice node is identified by an index `cI` stored in a Cartesian "frame" (box structured) for each grid level. In this way the computational time for all the successive operations, such as searching the neighbours and adding new points, becomes negligible because each node `cI` can be located by a singular index `i` encoding its coordinates in the "frame". In Listing 3.6, `rL` is the refinement level of the grid, `Nx_i`, `Ny_i`,



(a) (b) (c)

Figure 3.10: Overlapping grids stage operations. The grids are staggered by taking the cells centres (a – ■ symbol) as the lattice Boltzmann nodes (b). Finally the new interface nodes are created (c).

```
i = Nx_i + ( Ny_i * Nx_tot[rL] ) + ( Nz_i * Nx_tot[rL] * Ny_tot[rL] ) //i = frame index
//rL = refinement level
cI = frame[rL][i] //cI = node index
```

Listing 3.6: "frame" indexing.

and `Nz_i` specify the position of the node in X-, Y-, and Z-direction, and `Nx_tot`, `Ny_tot` (and also `Nz_tot`) are the total number of nodes in X-, Y- (and Z-) direction, respectively.

Creating the interface points

After staggering the grids, the next step is to create the interface. The cumulant LBM couples grid of different resolution by interpolation inside an overlapping region. The destination nodes of the interpolation in this region have to be generated. The interface consists of two sides, the coarse to fine (CF) and at the fine to coarse (FC). The number of layers of the new points created on each side is declared with the entries `createCoarse` and `createFine` in Listing 3.4. Figure 3.10c shows the added points at the interface (□ symbol).

Creating the solid nodes

The Eso-Twist data structure needs solid nodes as neighbours for those nodes that are at the border of the grid. Overlapping regions need solid nodes as well for those interpolation cells that intersect a boundary. Figure 3.11 shows both the cases. The first case is shown in horizontal direction. Eso-Twist has the advantage to require solid nodes only in positive directions. The second case is shown in vertical direction.

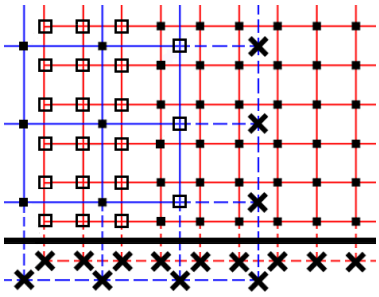


Figure 3.11: Creating the solid nodes. The Eso-Twist data structure requires an existing neighbour for each fluid node in positive X-, Y-, and Z-direction. In addition, the grid coupling requires complete interpolation cells, even if some of their nodes are solid. To obtain an admissible grid solid nodes (× symbol) are added where necessary.

The destination cells for the interpolation on the CF side need to be complete such that solid nodes in negative vertical direction must be generated. Setting the entry `createSolid` to `auto` tells `LBMHexMesh` to determine automatically in which directions to create solid nodes. If solid nodes are also necessary in negative directions (e.g. for tracking particles simulations), it is possible to force the code to create them by using the entry `createSolid` as `all`.

Renumbering the nodes

The next operation of the overlapping grid stage is the renumbering of the nodes. This is done only when `renumberMesh` is set to `true`. The cumulant LBM on GPGPUs runs faster with renumbered grids. The renumbering operation reorders the grid separately for each refinement level, from the lower corner to the upper corner of each grid, by proceeding first in X-, then in Y-, and finally in Z-direction.

Selecting the grid interface nodes

The second to last step is the selection of the grid interface nodes necessary for the grid interpolation. This is done by picking the lower corners of each source and destination cell (thick arrows in Figure 3.12). All nodes of the cell can be found starting from the lower corner by using the Eso-Twist pointer chasing algorithm. In order to have a valid interpolation on both sides of the interface, all nodes of the source cell must be fluid. However, where the grid interface crosses a boundary (thick black line) this condition is not satisfied. In this case an extrapolation is carried out from a neighbouring source cell. The distance of the lower corner of the extrapolation source cell to the lower corner of the invalid interpolation source cell (the offset) is required for computing the

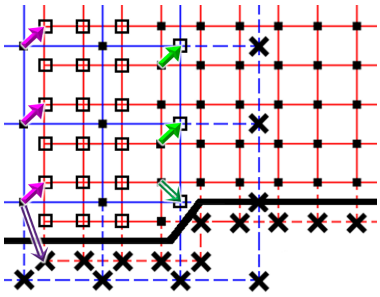
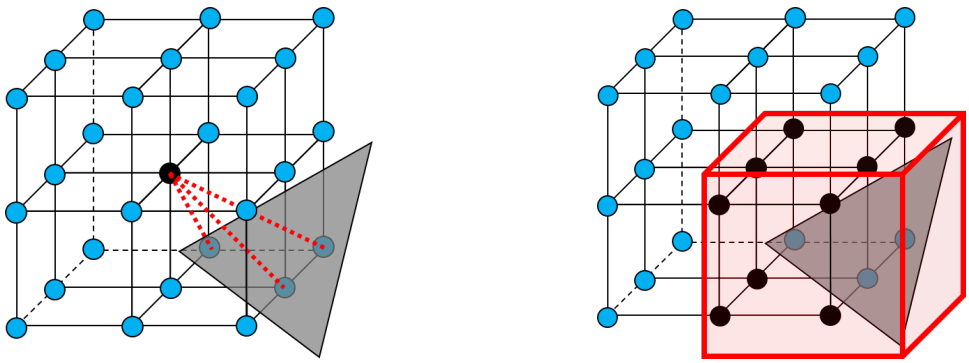


Figure 3.12: Selecting the grid interface nodes. The connections between the source cells and the destination cells for the interpolation are specified by the nodes at the lower corners of each cell (shown by arrows). If the interface crosses a boundary the source cells are not valid, and an offset vector has to be specified (double line arrows).

extrapolation coefficients at runtime (double line arrows in Figure 3.12). The extrapolation source cell is searched first in positive and negative X-, Y-, and Z-directions, and then in all the other diagonal directions. If no valid extrapolation source cell is found, the destination cell is removed from the fluid domain and all its nodes become solids. By using the "frame" matrix, the operations for searching the interpolation lower corners (both source and destination) are cheap. With the positions of the desired nodes known, picking the node indices on different levels requires no searching.

Computing the sub-grid-distances

The final operation of the overlapping grids stage is the computation of the sub-grid-distances, in order to have a second order accurate boundary definition. Considering the D3Q27 lattice, the sub-grid-distance q is computed for each lattice link that intersects a geometry. In the example of Figure 3.13a the lattice intersects a geometry represented by a triangle. The node at the centre of the lattice (in black) misses three fluid neighbours in the directions of the intersection (thick dashed line) and there qs have to be computed. The computation is carried out by using a ray-surface or a ray-triangle intersection technique [74] for basic surfaces (defined by the `searchableSurfaces` library) or for STL geometries, respectively. In order to do the computation faster, `LBMHexMesh` implements a bounding box technique that check only the nodes in the proximity of the geometry. If the geometries are defined by STL files, the algorithm goes through the STLs list, and it bounds each triangle by a box. Only the links from a fluid node inside the bounding box to a missing or solid node are checked for



(a) (b)

Figure 3.13: Computing the sub-grid-distances. The sub-grid-distances are computed by ray-surface intersection from a fluid node (in black) to the geometry (a). In order to perform the computation faster, the algorithm uses a bounding box of the geometry to reduce the number of nodes to be checked (b).

the link-triangle intersection (Figure 3.13b).If the geometry is defined by using the `searchableSurfaces` library, the algorithm bounds the overall geometry and proceeds similarly. `LBMHexMesh` includes corrections for fixing singularities (e.g. at the corners), multiple intersections (e.g. at very narrow features), and those nodes that were fluid and became solid (e.g. when no valid extrapolation source cell was found).

In order to control the computation of sub-grid-distances, it is possible to set four entries in the dictionary (they are the last four in Listing 3.5).

The first one is `assignBoundary` and it specifies how to assign the boundaries for the overall domain. It can be either `blockMeshly` or `geometrically`. The option `blockMeshly` assigns the boundaries following the OpenFOAM labelling, as declared in the background mesh dictionary. The option `geometrically` forces the code to set six boundary faces, one for each external face of the final grid bounding box. The discretized geometries are collected irrespectively of the `assignBoundary` option into a single boundary called `geom`.

The second entry is `exportWallNormals` and if `true` it allows to export the normal vectors for each boundary.

The third entry `setBcFixAsGeom` allows to choose whether to assign the nodes with corrected qs into the `geom` boundary (`true`) or keep them separated into a new boundary `fix` (`false`).

The last entry is `pressureCheck` and this is specific for the cumulant LBM implementation. Both pressure and extrapolation boundary conditions need at least one fluid neighbour in the direction normal to the boundary face. If set to `true`, the grid generator checks that all the nodes for all the boundaries are valid for applying a pressure condition. If no neighbour is found the node is set to solid.

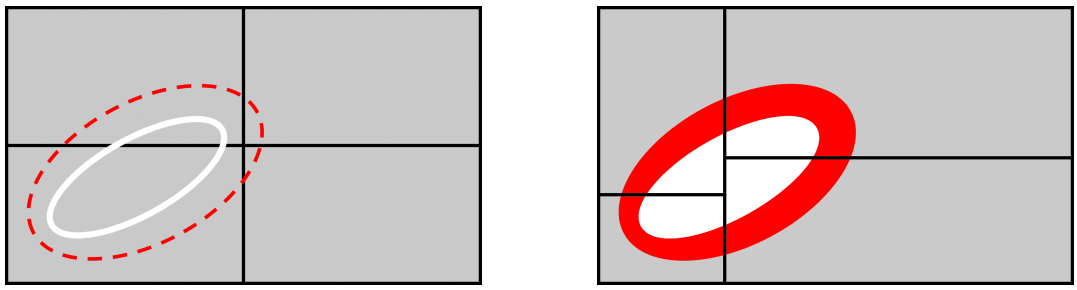
In the LBM a boundary specification requires both the node index and the sub-grid-distance q. Many boundary conditions (e.g. no-slip and velocity) are set for individual links, such that the same node can be part of different boundaries. When one node belongs to several boundaries at the same time, which link belongs to which boundaries depends on the type of the respective boundary conditions. For this reason, the node assignment and the sub-grid-distance assignment to the boundaries are separated. `LBMHexMesh` assigns the node indices to the boundaries and computes the sub-grid-distances, but it does not assign the qs to the boundaries. The assignment of the qs is done by a converter that reads the mesh generated by `LBMHexMesh` and sets the qs according to the boundary conditions specification. In this way the mesh created by `LBMHexMesh` is general and does not depend on the type of the boundary conditions.

```
numberOfSubdomains 4;
method hierarchicalLBM;
hierarchicalLBMCoeffs
{
    n ( 2 2 1 ); //nx * ny * nz = numberOfSubdomains
    delta 1E-16;
}
```

Listing 3.7: `decomposeParMeshDict` file example.

3.2.6 Parallel mesh generation

`LBMHexMesh` is a parallel grid generator producing also appropriate grids to run in parallel on multiple GPGPUs. Like `snappyHexMesh`, it starts from a previously decomposed background mesh. The decomposition is performed by using the OpenFOAM application `decomposePar`, after setting the parameters into the dictionary file `decomposeParMeshDict` (Listing 3.7). The entries of the dictionary are: i) the total number of divisions (or processors – `numberOfSubdomains`), that corresponds to the number of GPGPUs used for the simulation, ii) the decomposition method that must be `hierarchicalLBM`, and iii) the number of divisions in X-, Y-, and Z-direction (`n`), taking into account that $nX \times nY \times nZ$ must be equal to the total number of divisions. `LBMHexMesh` works only with planar interfaces between processors. For this reason, `LBMHexMesh` introduces the new decomposition method `hierarchicalLBM` that splits the background mesh into different parts with planar interfaces (Figure 3.14a). This was necessary because the original `hierarchical` decomposition method does not satisfy the planarity of the interfaces, especially if the background mesh has, in a certain direction, a number of cells not divisible by the set



(a) (b)

Figure 3.14: Load balancing. The decomposition of the background mesh produces the same number of sub-grids as the number of processors/GPGPUs used (a), and the processor interfaces are planar. During the refinement stage the level wise load balancing equalizes the number of points for each grid level and for each processor by keeping the planarity of the interfaces (b).

number of divisions.

Load balancing

In the cumulant LBM a coarse grid time-step is equivalent to two fine grid time-steps. Therefore, in order to have the same number of operations on each GPGPU, the number of nodes for each refinement level on each GPGPU should be the same. For this reason `LBMHexMesh` implements a new load balancing algorithm that complies with this constrain. The algorithm optimizes the balancing between different grid resolutions performing a level wise load balancing, keeping also the processor interfaces planar (Figure 3.14b). Furthermore, since the grid refinement can also cross the processor interfaces, the algorithm checks whether each communication node on every processor has the respective duplicated node on the other processor. The algorithm works similarly to that one implemented to check the width of the grid interface (Figure 3.8). The load balancing acts only on the refinement stage, and it can be activated by using the entry `loadBalance` in the dictionary `LBMHexMeshDict` (Listing 3.4). If the geometry is symmetric, `loadBalance` can be switched to `false`.

Creating the send and receive nodes

The communication between GPGPUs is performed by using duplicated nodes ("receive" nodes in Figure 3.4). Like the nodes used for building the grid interface or the solid nodes, these nodes are created during the second stage of the grid generation, after the load balancing. The indices of the duplicated nodes are exchanged once before the simulation via OpenMPI [75]. The use of the "frame" facilitates the finding of those indices.

3.2.7 Setting the case

The assignment of the sub-grid-distances to the boundaries is done independently from the mesh generation. During this operation, the values for the boundary fields, such as pressure and velocity, are set and the simulation can be started. In order to prepare the case, it is necessary to define the boundary conditions into a new dictionary file `LBMCaseSetupDict` as shown in Listing 3.8. This file declares for each boundary which field has to be set (`field`), the type of the boundary condition (`type`), and the value (`value`).

The possible `field` options are: `pressure`, `outflow`, `velocity`, `noSlip`, `slip`, and `periodic`.

```
inlet   { field velocity; type uniform; value ( 16 0 0 );}
outlet  { field outflow; type uniform; value  0;}
front   { field velocity; type uniform; value ( 16 0 0 );}
back    { field velocity; type uniform; value ( 16 0 0 );}
bottom  { field velocity; type uniform; value ( 16 0 0 );}
top     { field velocity; type uniform; value ( 16 0 0 );}
geom          { field velocity; type uniform; value ( 0 0 0 );}
geom_wheelsF { field velocity; type rotatingWall; value ( ( 0 0 0 ) 50.1 -y );}
geom_wheelsR { field velocity; type rotatingWall; value ( ( 2.8 0 0 ) 50.1 -y );}
```

Listing 3.8: `LBMCaseSetupDict` file example.

The boundary condition `type` options depend on the `field`. For all the fields the standard entry is `uniform`, imposing a fixed uniform value. For `velocity` it is possible to set a parabolic profile (three-dimensional `parabolic3D`, circular `parabolicCirc`, and two-dimensional `parabolic2D`), a tangential rotating velocity (`rotatingWall`), and the atmospheric boundary layer profile (vertical `atmBoundaryLayer` and circular `atmBoundaryLayerCirc`).

The options for the `value` entry depend on both the field and the type chosen. In the example of Listing 3.8 (that refers to the car model already shown in Figure 3.7), a fixed uniform velocity of 16 m/s is used in X-direction for the inlet and all the sides of the tunnel, including the ground (`bottom`) to simulate the moving road. For the outlet the boundary condition is `outflow` that sets a zero gradient for the velocity by using extrapolation. The car (`geom`) is a wall with velocity 0 m/s and, after dividing the STL file in different parts, it is possible to specify different boundary types for some of them. For example, the wheels (front `geom_wheelsF` and rear `geom_wheelsR`) rotate by using the `rotatingWall` type. The `value` entry for this boundary is composed of the centre of rotation, the rotational velocity ($1/s$), and the direction of rotation.

3.2.8 Exporting the grid

The dictionary for preparing the case (Listing 3.8) can be executed with two different applications implemented to export the grid. The first one is `LBMFoamToVTK`, which converts the cumulant LBM grid into Visualization Toolkit (VTK) format [76] in order to visualize it, e.g. with the open-source post-processing software ParaView [77]. The second one is `LBMFoamToIRMBGPU`, which writes the mesh in the specific format necessary for performing simulations on the GPGPU. All the converted files can be either in ASCII or binary format.

Particular attention was paid to write all the files in an optimized format. Listing 3.9 shows an example for a single line of the sub-grid-distance file. The line reports

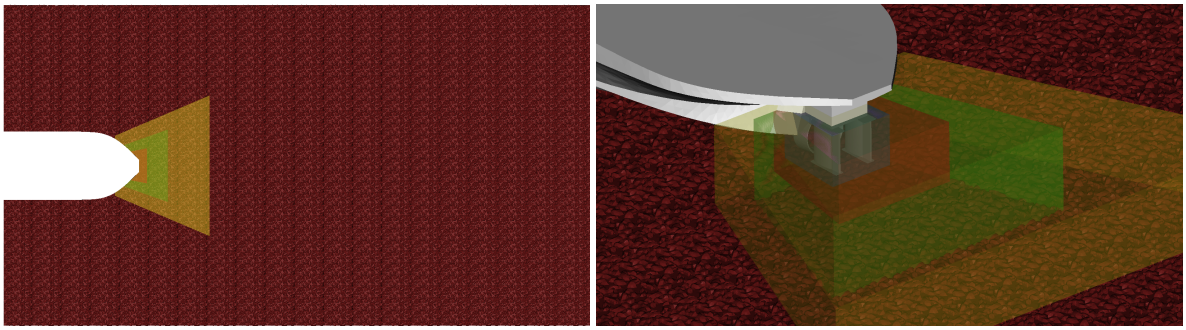
```
240 2149461 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5
//  2149461 (integer) = 10101010001100110000010000 (binary)
//  - binary: 27 bits for the 27 links of the D3Q27 lattice
//  - links: from direction 26 (first link) to direction 0 (last link)
//  - only links with symbol "1" have sub-grid-distance
```

Listing 3.9: sub-grid-distance file example.

the index of the node (first number), the directions of the qs (second number), and the values of the qs (in this case they are all 0.5). The directions are written by using only a single number, which is the decimal conversion of the complete binary list of the 27 links of the D3Q27 lattice. By reconverting the directions number in its binary form it is possible to find in which link the node has a sub-grid-distance. The value `1` means that the link has the q, while the value `0` means that the link does not have any q. Each link index refers to a specific lattice direction, e.g. position 0 is (+X) and position 26 is (-Z,-Y,-X). The order for reading the binary number is from the last link index (26) to the first one (0). Hence, the first q value is related to the position of the first `1` in the binary number, the second q to the position of the second `1`, and so forth. In this way the amount of space for writing the sub-grid-distances is considerable reduced in comparison to defining all the qs for each link.

3.3 Results

This section presents some cases of grids generated with `LBMHexMesh`, both in serial to simulate on a single GPGPU and in parallel for multi-GPGPUs applications.

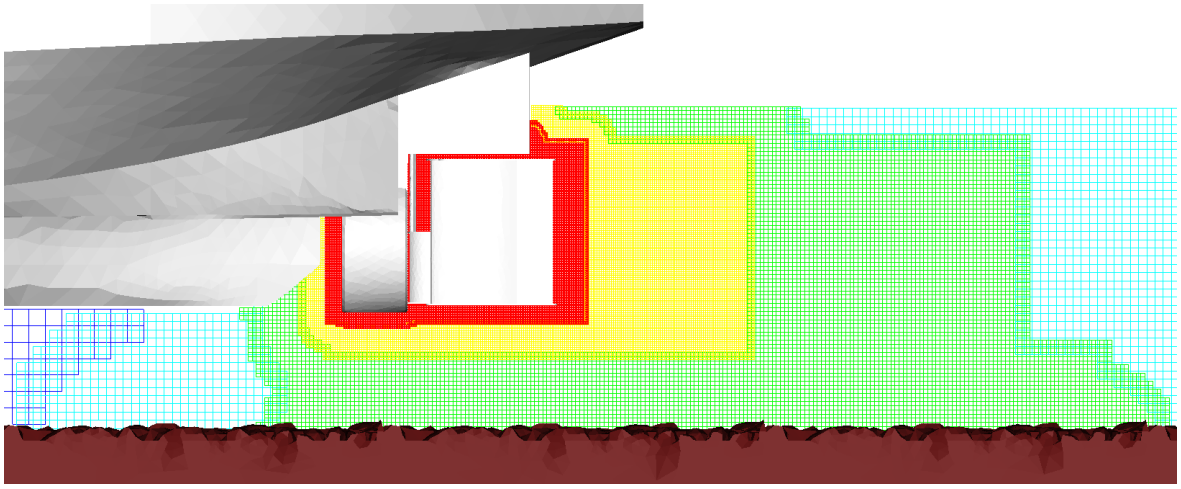


(a) Top view. (b) Isometric view.

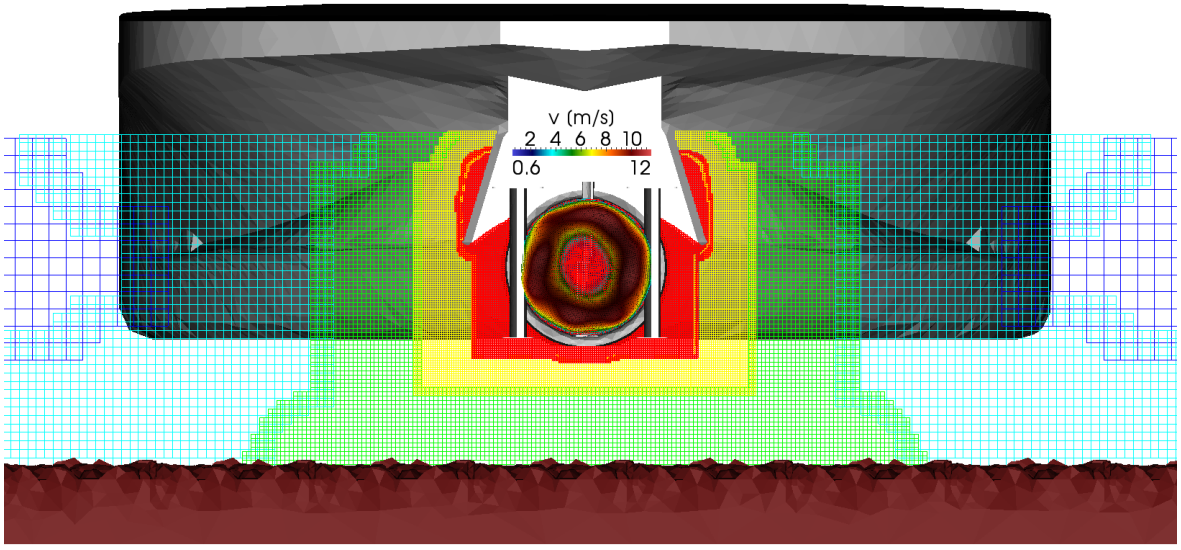
Figure 3.15: Ship over a river – geometry. The ship (in light-grey) sails over a river with a detailed bed (in brown). The coloured boxes are used as refinement regions.

3.3.1 Ship over a river

The first case analysed was a ship sailing over a river. The geometries consisted of a ship, the bed of the river, and four additional boxes used as refinement regions (all are STL files – Figure 3.15). The background mesh was generated in order to simulate the top of the domain as the free surface of the river, cutting the ship at a certain height. The final mesh was of 23 586 786 points consisting of 21 538 323 fluid and

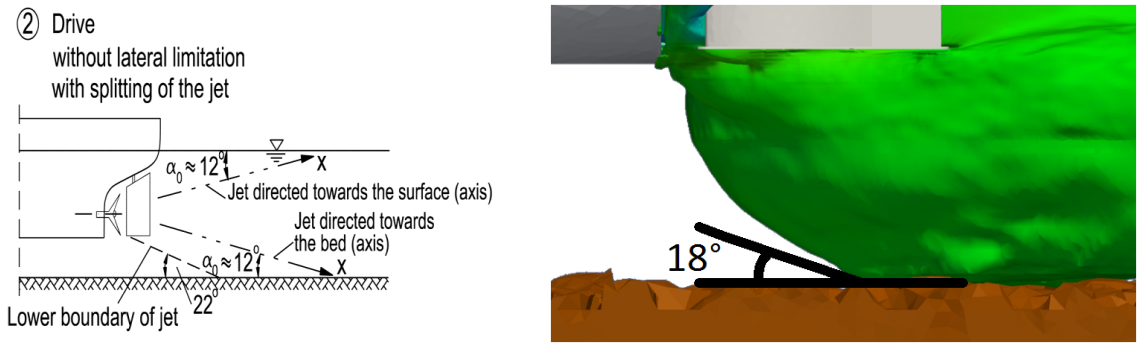


(a) Side view.



(b) Front view.

Figure 3.16: Ship over a river – grid. The colours of the grid wire-frame are the refinement levels. The motion of the propeller is modelled by a velocity profile (b).



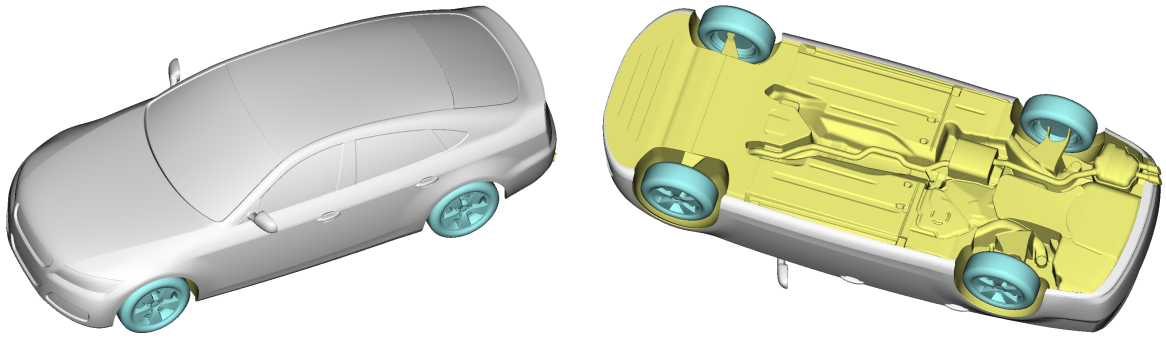
(a) (b)

Figure 3.17: Ship over a river – results. Jet dispersion angle as stated by the BAW [78] (a) and the numerical result from an isosurface at mean velocity of 1.2 m/s [68] (b).

2 048 463 solid nodes. The grid had five refinement levels with cell size $\Delta x_0 = 0.2\ m$ for the background mesh and $\Delta x_4 = 0.0125\ m$ for the refinement box surrounding the propeller. The realization of the grid interfaces was particularly difficult because the grids interface intersected the ship geometry on every grid level, requiring offset vectors for both fine to coarse (FC) and coarse to fine (CF) sides (Figure 3.16). The grid interface was defined with 190 404 (1 100 with offset) and with 201 046 (4 070 with offset) connections for the FC and CF sides, respectively. The two geometries were discretized with 888 888 boundary nodes. The grid was generated in 55 min CPU time, consisting of 39 min and 17 min for refining the grid and building the overlapping regions, respectively.

The boundary conditions were fixed pressure for the two longitudinal boundaries, slip for the top and the two sides, and no-slip for all the geometries. The motion of the fluid was provided by imposing a velocity profile that simulated the rotation of the propeller (Figure 3.16b). The fluid was water and the Reynolds number simulated was of 31.1×10^6 (based on the propeller diameter reference length).

The simulation ran on a single NVIDIA GeForce GTX TITAN GPGPU in single precision, for 300 s real time in less than 24 h [68]. Figure 3.17 shows the result for the jet dispersion angle, which was calculated by considering an isosurface for a mean velocity of 1.2 m/s [68]. The angle found was of 18° and it was in line with that one of 22° stated by the German Federal Waterways Engineering and Research Institute (Bundesanstalt für Wasserbau – BAW) [78].



(a) Top view. (b) Bottom view.

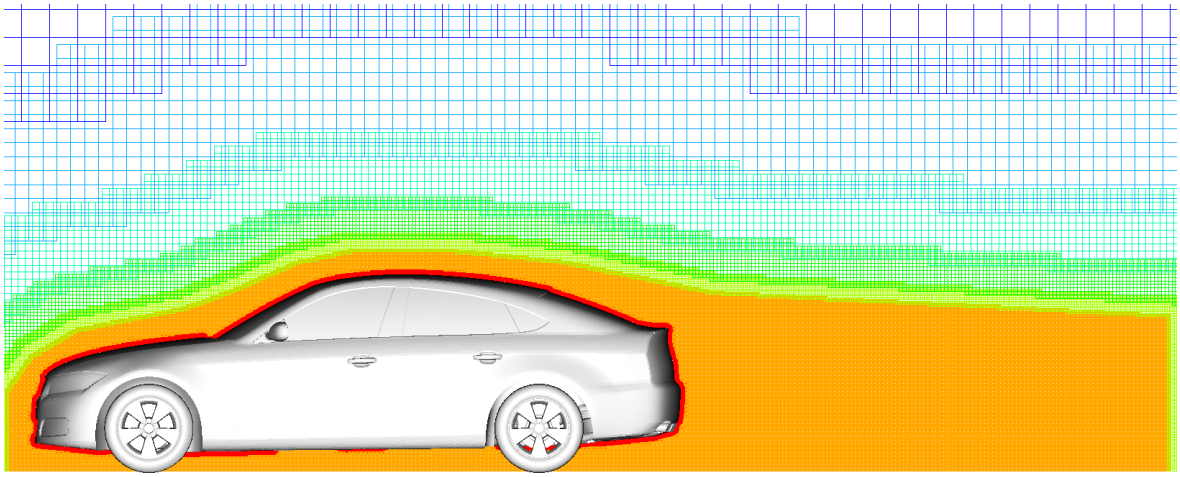
Figure 3.18: Car model – geometry. The car is the DrivAer model [79] in its fastback body configuration, with side mirrors, wheels, and detailed underbody.

3.3.2 Car model

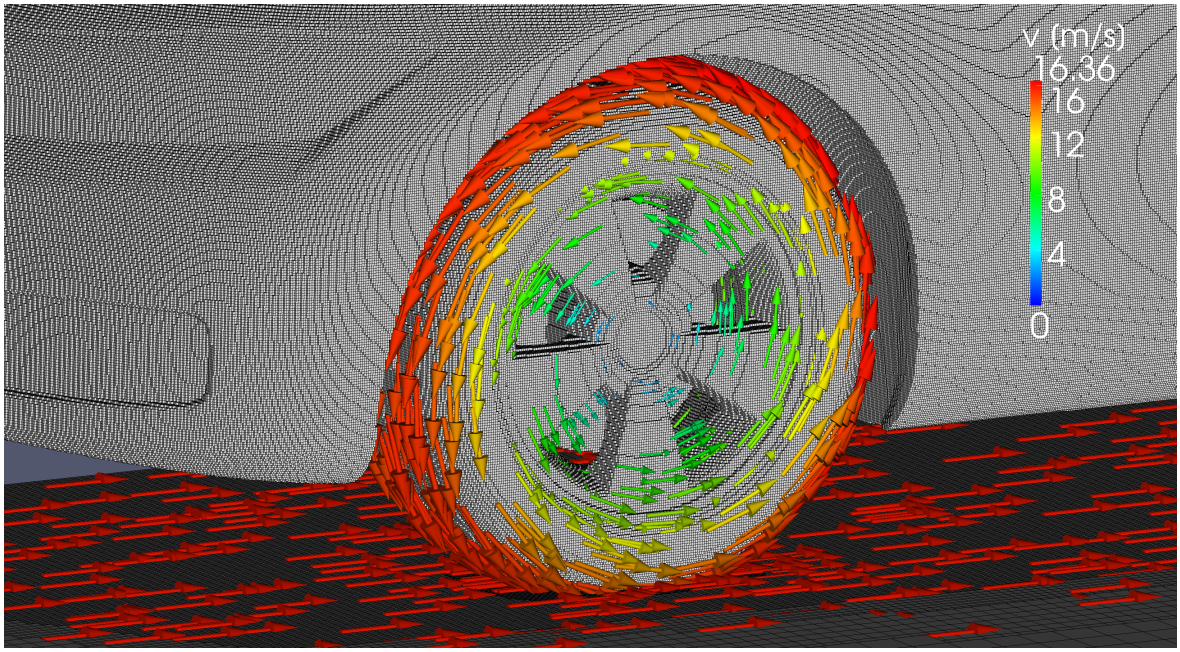
The second case was the car used as an example in the previous section. The car geometry was created by the Technische Universität München [79] and was discretized with more than 10^6 triangles by using the pre-processing software ANSA [80]. The blockage ration was about 0.8%. Figure 3.18 shows the car geometry for the fastback body configuration, with side mirrors, wheels, and detailed underbody. In order to perform a grid converge study, three grids with different resolutions were generated. The finer grid was generated in parallel for running on two GPGPUs, and the domain was perfectly cut at the symmetry plane of the car (Figure 3.19a). The final mesh had 125 580 962 points consisting of 116 353 891 fluid and 9 227 071 solid nodes. The grid had seven refinement levels with cell size $\Delta x_0 = 0.2\ m$ for the background mesh and $\Delta x_6 = 0.003125\ m$ for the car, producing a discretization of the geometry with 4 484 952 boundary nodes. The refinement region for the wake was of level five with $\Delta x_5 = 0.00625\ m$. The interfaces between the two sub-domains for the GPGPUs communication had 270 070 nodes spread over all the refinement levels. The grid was generated in 189 *min* CPU time, consisting of 70 *min* and 119 *min* for refining the grid and building the overlapping regions, respectively.

The boundary conditions were outflow for the outlet and fixed velocity for the inlet and all the other planes, including the ground for simulating the moving road. The car was set with a no-slip condition for the body and a tangential velocity for simulating the rotation of the wheels (3.19b). The fluid was air and the Reynolds number simulated was of 4.87×10^6 (based on the total car length).

The simulation was carried out on two NVIDIA Tesla K40c GPGPUs in single precision. The complete simulation of 5 *s* real time was performed in 63.4 *h* [67].



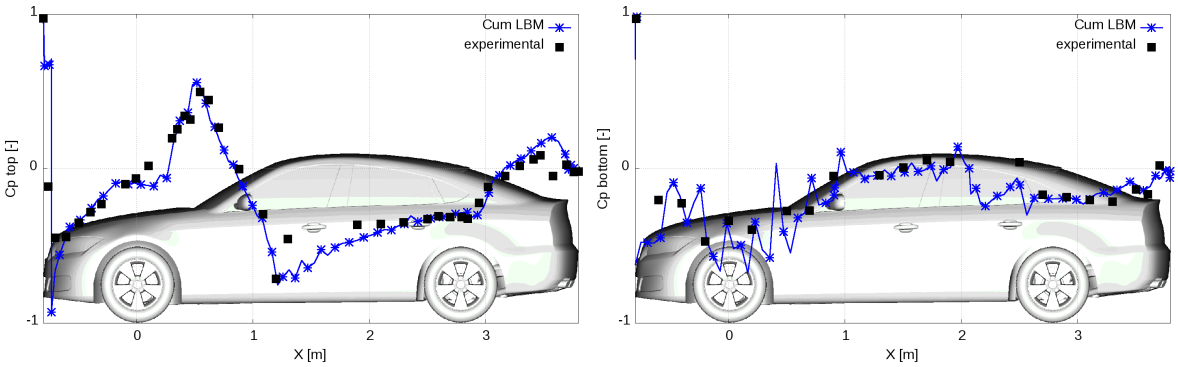
(a) Side view.



(b) Wheel detail.

Figure 3.19: Car model – grid. The colours of the grid wire-frame indicate the refinement levels in the top picture (a), while in the bottom picture they indicate the velocity of the vectors for the boundary conditions used for the moving ground and the rotation of the wheels (b).

Figure 3.20 shows the results for the pressure coefficient over the car symmetry plane for the top (Figure 3.20a) and bottom part (Figure 3.20b), respectively. They had a good agreement with the experimental measurements. Moreover, the drag coefficient was found to be 0.274 compared to the experimental value of 0.275.

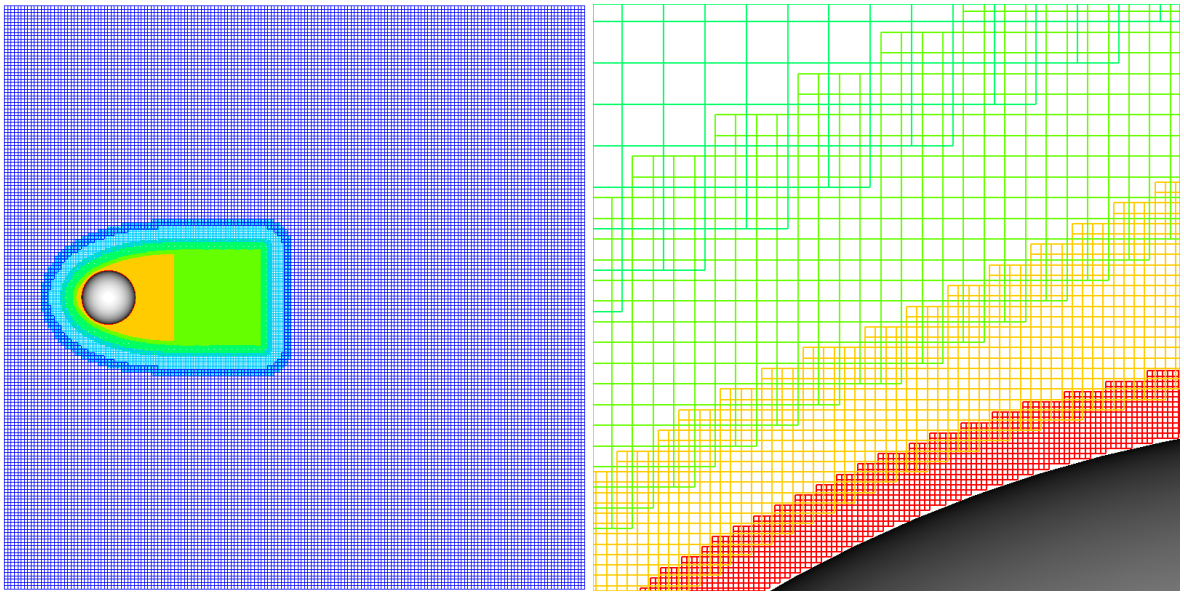


(a) Top. (b) Bottom.

Figure 3.20: Car model – results. Pressure coefficient along the symmetry plane of the car for the top part (a) and the bottom part (b).

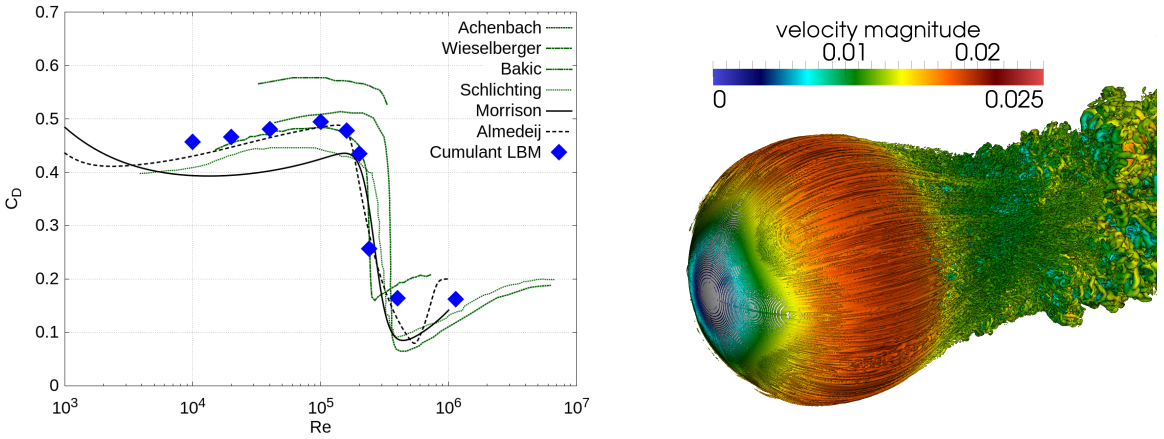
3.3.3 Flow past a sphere

The third case analysed was the flow past a sphere. The grid was generated in parallel for running on four GPGPUs. The domain was perfectly cut at the two symmetry planes of the sphere in Y- and Z-direction (Figure 3.21). The background mesh was a cube with the blockage ratio of 0.65%. The final mesh had 73 855 027 points consisting of 70 763 711 fluid and 3 091 316 solid nodes. The grid had six refinement levels in order



(a) Side view. (b) Detail.

Figure 3.21: Flow past a sphere – grid. Side view (a) and a closer look at the geometry surface (b) for the sphere discretization. The colours indicate the grid refinement levels.



(a) (b)

Figure 3.22: Flow past a sphere – results. Drag coefficient for different Reynolds numbers (a) and an isosurface for the Q-criterion at $Re = 1.14 \times 10^6$ (b – the velocity is in lattice units).

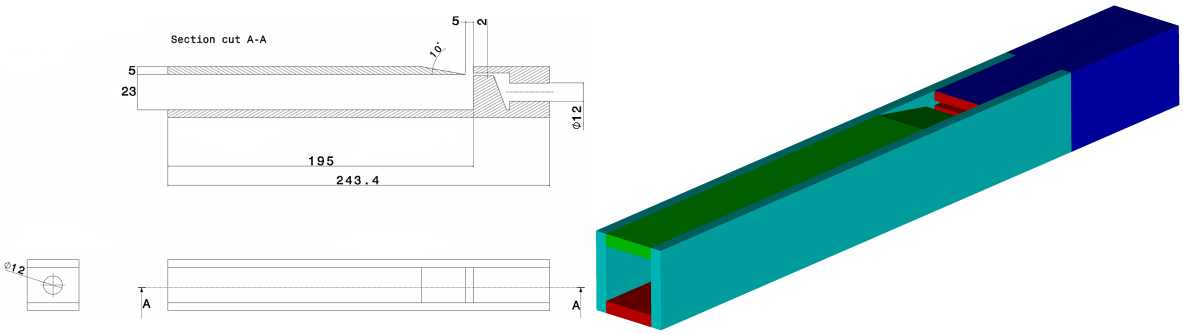
to cover the sphere diameter with 512 nodes, producing 1 241 504 boundary nodes for the whole geometry discretization. For resolving the wake, two refinement regions were used (bullet shaped – Figure 3.21a). They had level four and three for the shorter and the longer one, respectively. The interfaces between the sub-domains for the GPGPUs communication had about 120 000 nodes for each side spread over all the refinement levels. The grid was generated in 173 *min* CPU time, consisting of 131 *min* and 42 *min* for refining the grid and building the overlapping regions, respectively.

The boundary conditions were outflow for the outlet, fixed velocity for the inlet and all the other planes, and no-slip for the sphere surface. The Reynolds numbers simulated were in a range from 10 000 to 1.14×10^6 (based on the sphere diameter).

The simulations were carried out on four NVIDIA Tesla C1060 GPGPUs in single precision. Figure 3.22a shows the results for the drag coefficient for different Reynolds numbers. It had a good agreement with the experimental data [81–83] and regression functions [84, 85], demonstrating the capability to capture the drag crisis. Moreover, an isosurface for the Q-criterion [86] is given in Figure 3.22b for the $Re = 1.14 \times 10^6$. There, the eddies close to the sphere had a fine structure.

3.3.4 Organ pipe

The fourth case was an organ pipe. The model has been created for a student specialization project [87], and the pipe, designed by using the CAD software CATIA [88] (Figure 3.23a), has been first discretized in STL format with ANSA [80] (Figure 3.23b) and then physically created by using a 3D printer. In order to have experimental



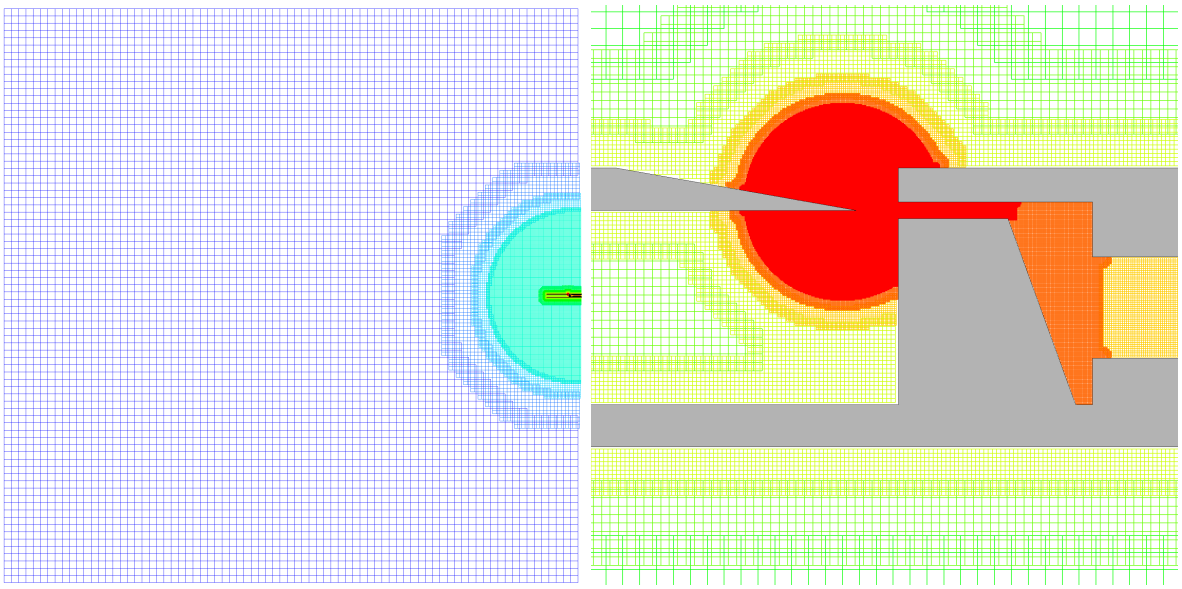
(a) CAD design.

(b) STL geometry.

Figure 3.23: Organ pipe – geometry. The design of the organ pipe (a – dimensions in mm) and its representation in STL format (b).

data for the comparison with the numerical results, the real model has been tested by the National Metrology Institute of Germany (Physikalisch-Technische Bundesanstalt – PTB) [89].

The background mesh was large enough in order to avoid any reflections from the external boundaries, and had a resolution of $\Delta x_0 = 0.064\ m$ (Figure 3.24a). The grid had 11 refinement levels in total and the finest resolution was $\Delta x_{10} = 6.25 \times 10^{-5}\ m$ at the pipe separation edge. There, in order to have a larger area with that resolution, a



(a) Slice at the symmetry plane.

(b) Detail.

Figure 3.24: Organ pipe – grid. Slice at the symmetry plane of the simulation domain (a) and its detail at the pipe edge (b). The colours indicate the grid refinement levels.

spherical refinement region was used (Figure 3.24b). In order to capture the pressure waves with an adequate resolution, another spherical refinement region of level three ($\Delta x_3 = 0.008\ m$) surrounded the complete pipe. The mesh ensured a smooth transition between different resolutions by using four cells between grid interfaces. The final mesh has been generated in parallel on two processors producing 43 233 148 grid points, consisting of 40 052 257 fluid and 3 180 891 solid nodes, and the pipe geometry was discretized with 945 016 boundary nodes. Due to the changing of resolution on the pipe surface, the grid had 402 and 5 150 offset vectors for the FC and CF sides, respectively. The grid was generated in 33 *min* CPU time, consisting of 19 *min* and 13 *min* for refining the grid and building the overlapping regions, respectively.

The boundary conditions were fixed uniform velocity at the entrance of the pipe, outflow at the outlet of the domain, and no-slip condition for the pipe and the external planes of the domain. The fluid was air and the inflow velocity has been tested in the range $3 - 14\ m/s$.

The grid computed on two NVIDIA Tesla K40c GPGPUs in single precision for 20 000 iterations corresponding to more than 2 s real time. Figure 3.25 shows the results for the sound spectrum for a measurement point located at 50 mm in front of the pipe with the tested velocity of 3 m/s and 4 m/s. Unfortunately, the exact velocity at the inflow could not be determined in the experiment, such that this comparison had a large and unknown experimental uncertainty. Still, the reasonable agreement between the numerical and experimental data demonstrated the possibility to perform CFD simulations for acoustics problems with the cumulant LBM.

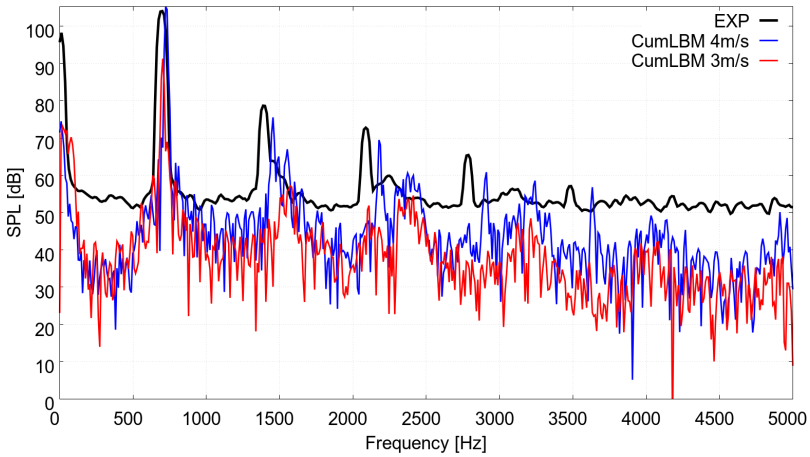
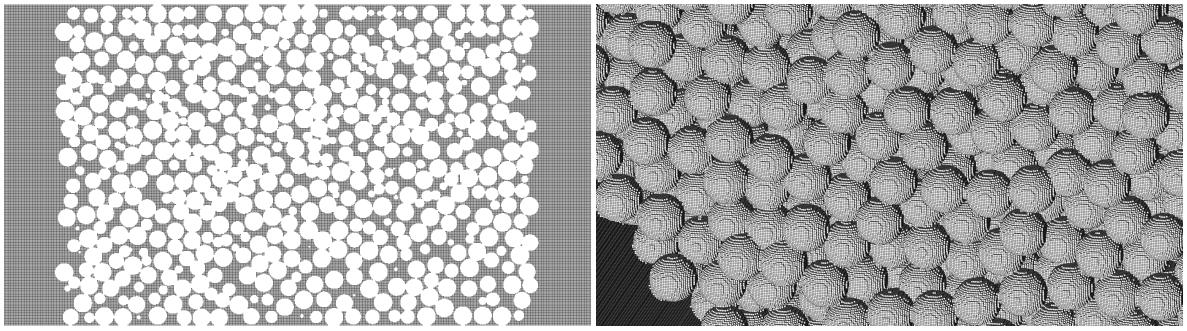


Figure 3.25: Organ pipe – results. Sound spectrum for a measurement point located at 50 mm in front of the pipe.



(a) Side view. (b) Detail.

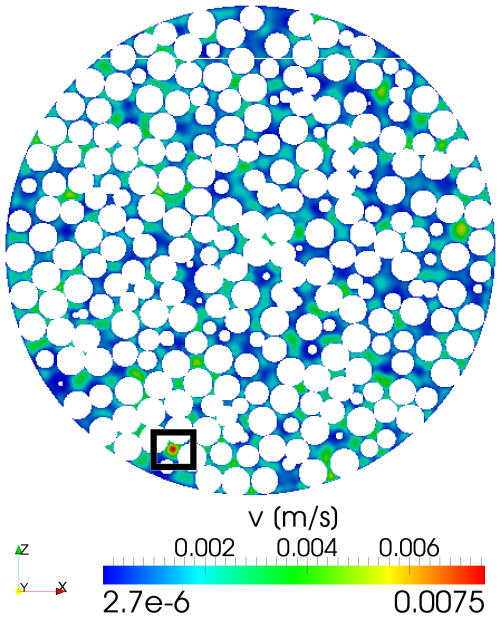
Figure 3.26: Porous medium – grid. Longitudinal slice in the middle of the pack of spheres (a) and a closer look at the particles (b) as discretized by using `LBMHexMesh`.

3.3.5 Porous medium

The second to last case was about the porous media fluid flow and solute transport at the pore scale [90]. The porous medium was composed of more than 6 800 spherical particles and the geometry of each particle and the surrounding cylindrical column were defined by using the `searchableSurfaces` OpenFOAM library. Two uniform grids with different resolutions were generated ($\Delta x = 20\ \mu m$ and $40\ \mu m$). The meshes were generated to run either in serial, for the coarser resolution, or in parallel on two or six GPGPUs, for the finer resolution (Figure 3.26a). In the latter case, the level wise load balancing equalized the number of grid points on each processor. The final mesh for the $\Delta x = 20\ \mu m$ resolution had 82 108 430 points consisting of 71 815 476 fluid and 10 292 954 solid nodes, discretizing the geometries with 20 737 803 boundary nodes (Figure 3.26b). The grid was generated in 13 h CPU time, consisting of 9 h and 4 h for refining the grid and building the overlapping regions, respectively.

The boundary conditions were fixed mass flow at the inlet (by setting an uniform velocity), fixed pressure at the outlet, and no-slip condition for the particles and the column wall. The fluid was water and the Reynolds number simulated was of 0.6 (grain based).

The simulation was carried out on two NVIDIA Tesla K40c GPGPUs in single precision and it performed 72 s of real time in 61 h. Figure 3.27a shows the results for the velocity field mapped on a slice normal to the flow direction. There, the velocities for a sample of five points were selected (located in the black square) and Figure 3.27b [90] reports the values for the X-, Y-, and Z-components. The cumulant LBM results were comparable with those obtained with different simulation approaches. Moreover, the pressure drop along the column was calculated to be 16.07 Pa and it



(a)

	STAR-CCM+	TETHYS		LBM		ISPH	
		20μm	40μm	20μm	40μm	20μm	40μm
	v_y (mm/s)						
Point 1	7.142	7.038	6.622	7.261	5.288	6.847	6.035
Point 2	5.387	5.942	5.667	6.487	4.921	5.323	5.453
Point 3	5.186	5.013	4.774	4.034	2.788	5.191	5.213
Point 4	1.860	2.042	1.958	2.358	1.497	1.862	1.890
Point 5	0.425	0.431	0.407	1.298	1.100	0.424	0.432
	v_x (mm/s)						
Point 1	−0.550	−0.566	−0.584	−0.629	−0.447	−0.541	−0.562
Point 2	−0.542	−0.539	−0.514	−0.684	−0.505	−0.539	−0.548
Point 3	−0.636	−0.641	−0.622	−0.706	−0.424	−0.635	−0.642
Point 4	−0.096	−0.094	−0.102	−0.082	−0.059	−0.089	−0.092
Point 5	0.088	0.093	0.090	0.091	0.025	0.086	0.084
	v_z (mm/s)						
Point 1	−1.824	−1.852	−1.773	−1.975	−1.430	−1.814	−1.791
Point 2	−1.484	−1.430	−1.391	−1.624	−1.213	−1.464	−1.452
Point 3	−0.974	−1.103	−1.102	−1.116	−0.849	−0.945	−0.958
Point 4	−0.942	−0.931	−0.906	−0.683	−0.443	−0.941	−0.947
Point 5	−0.634	−0.592	−0.682	−0.858	−0.604	−0.641	−0.659

(b)

Figure 3.27: Porous medium – results. Contour plot of the velocity on a slice normal to the flow direction (a) and velocity values for a sample of five points (located in the black square) by using the cumulant LBM and other simulation approaches (b) [90].

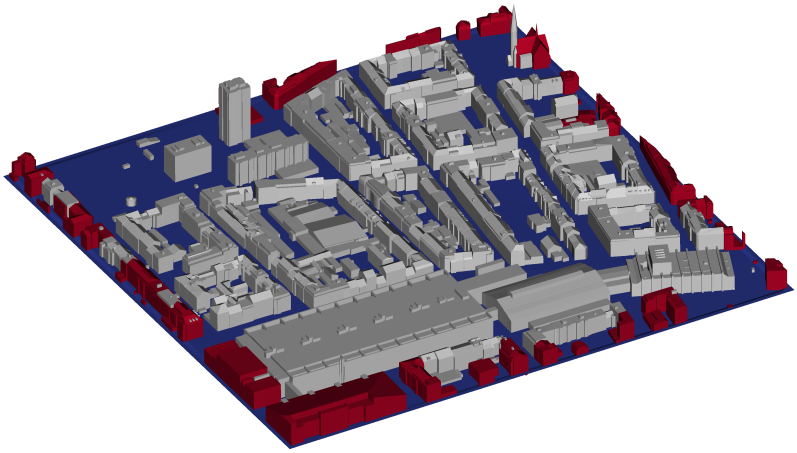
was in accordance with the literature.

3.3.6 Basel down town

The last case was about a turbulent flow over a real urban area. The urban area considered was the city of Basel, in Switzerland [91]. The geometry provided was in CAD format (Figure 3.28a). The file did not comply with the quality requirements of CFD, i.e. there were no closed edges and duplicated surfaces. Therefore, the file was manually repaired by using the software ANSA [80]. The complete geometry was divided in order to select a sub-domain of the Basel down town of dimensions 512 $m \times 512$ m (Figure 3.28b).

The mesh generation and the case setup for this model was conducted for a stu-

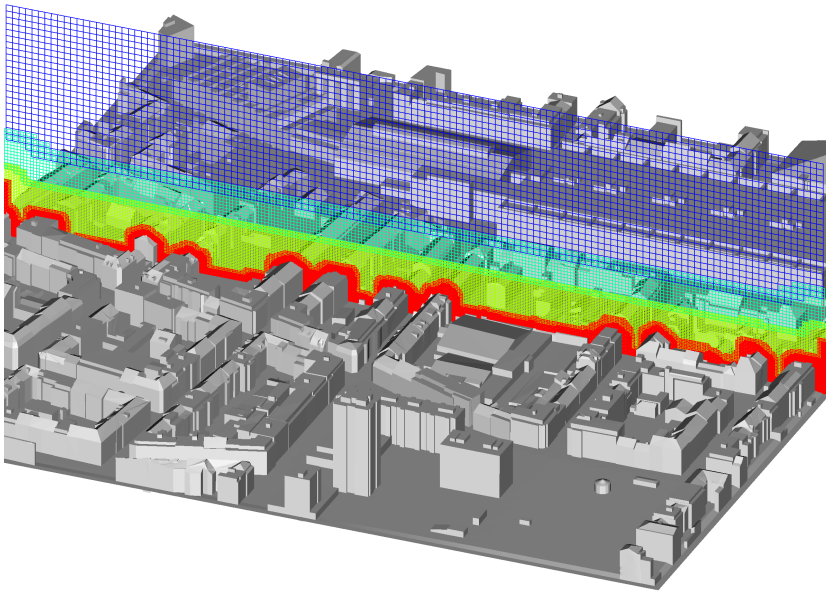
(a) Overall geometry.



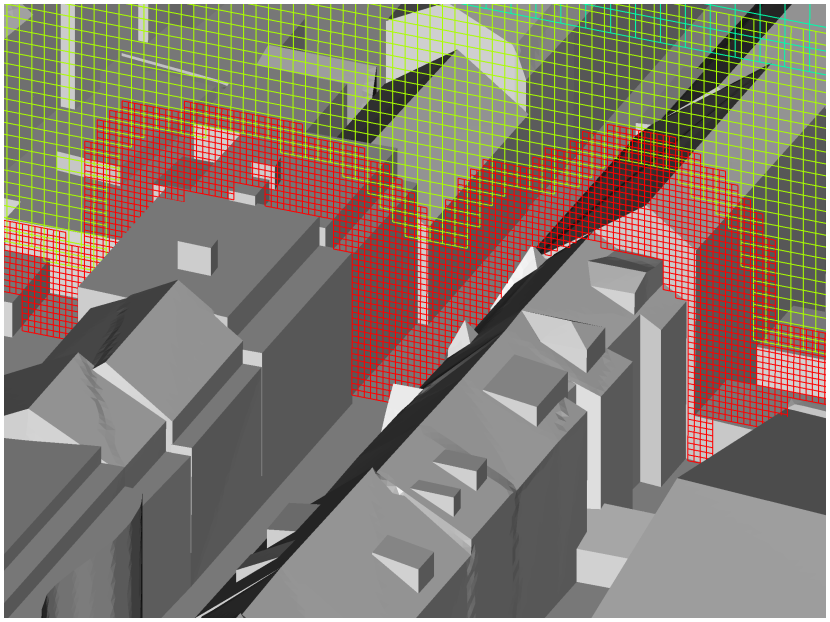
(b) Computed sub-domain geometry.

Figure 3.28: Basel down town – geometry. The complete city geometry in CAD format (a) and its final configuration for the grid generation in STL format (b).

dent specialization project [92, 93]. Three grids were constructed, a coarser one with resolution at the buildings of 1 m, a finer one with 0.5 m, and a uniform one with the resolution of 1 m for the whole domain. The first two grids had a background mesh of $\Delta x_0 = 4$ m resolution and all the meshes were generated in serial to run on a single



(a) Slice at the measurement tower location.



(b) Detail.

Figure 3.29: Basel down town – grid. Slice at the measurement tower location of the Basel down town model (a) and its detail (b). The colours indicate the grid refinement levels.

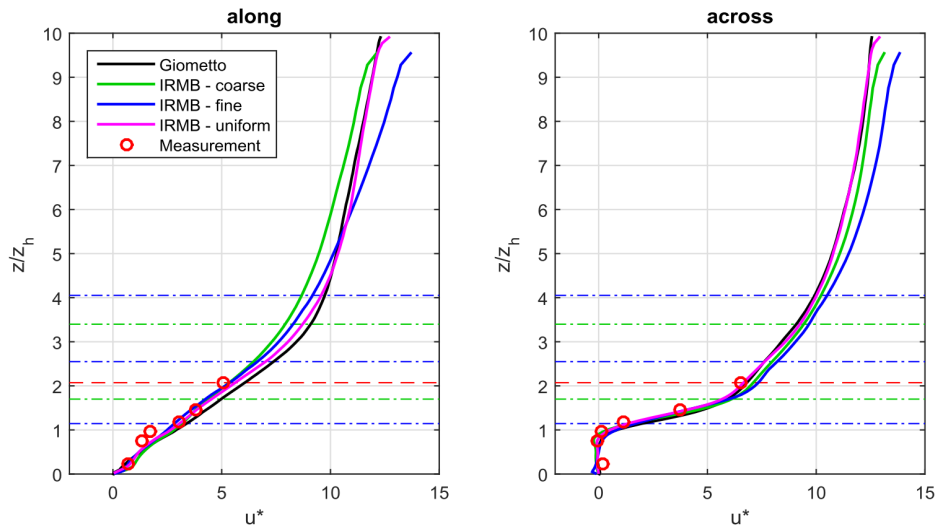


Figure 3.30: Basel down town – results. Velocity profile in stream-wise direction for a wind along (left) and across the street (right) [92].

GPGPU. The finer mesh was composed of four refinement levels, with 41 112 498 grid points consisting of 36 644 015 fluid and 4 468 483 solid nodes, and the city geometry was discretized with 2 344 672 boundary nodes (Figure 3.29). The grid was generated in 92 *min* CPU time, consisting of 44 *min* and 49 *min* for refining the grid and building the overlapping regions, respectively.

The boundary conditions were no-slip for the buildings and the ground, full slip for the top, while all the other lateral planes were periodic. The fluid was air and it was forced to move by a constant forcing simulating a certain pressure gradient along the stream-wise axis. The grid scale Reynolds number simulated was of 0.794×10^6 (based on the background mesh resolution of 4 m).

The finer grid case ran on a single NVIDIA Tesla K40c GPGPUs in single precision, performing circa 6 h of real time in 55 h [92]. Figure 3.30 shows the results for the velocity profile at the position of the measurement tower, for a wind along (left) and across the street (right) [92]. They demonstrated a good agreement with both the experimental data and the numerical results obtained with other approaches [91].

3.4 Discussion and summary

This Chapter addressed the discretization process of very complex geometries for the cumulant LBM implementation on the GPGPU [65, 68], leading to the development of a new LBM grid generator. `LBMHexMesh` produces free shape multi-level three-

dimensional grids with second order accurate boundary definition, for both serial and parallel simulations.

The discretization has three main aspects: i) The requirement of Cartesian grid does not simplify the mesh generation process, especially if grid refinement and second order boundary definition are necessary. ii) For parallel simulations, the load balancing between processors must be level wise because the time-step length depends on the grid resolution. iii) The definition of the grid properties must be as compact as possible in order to either save memory for the computation or use larger models with the same memory.

With regards to the first aspect, the grid refinement in `LBMHexMesh` is not shape-constrained and it is possible to select different refinement techniques depending on the desired configuration (body fitted, geometrical regions, external STL regions, distance mode, inside mode, etc.). The computational time for the definition of the grid interface is negligible compared to the time needed for the refinement stage. This was possible by using a block structured data arrangement only for the final LBM mesh ("frame" – Listing 3.6), while the refinement stage used a standard octree structure [72]. Regarding the boundary representation, `LBMHexMesh` respects the cut-cell approach [63] in order to have a second order discretization even with a Cartesian mesh.

Regarding the second aspect, the load balancing issue was addressed by implementing a new level wise technique. In this way it is possible to satisfy a balancing of the number of points for each processor for each grid level. Further work could be carried out in order to allow irregularly shaped communication interfaces (not only planar), by considering for example the METIS/ParMETIS libraries [94].

Regarding the last aspect, the compactness of the data was addressed by paying particular attention for GPGPUs applications in order to minimize the memory footprint on the GPGPU. For this reason, grid generation and simulation are two separated stages. In this way it is possible to allocate only the effective data necessary for executing the cumulant LBM kernel on the GPGPU. All the other information, e.g. the geometries files, is never required by the kernel and the amount of data on the GPGPU is minimized. In addition, by using innovative data structure like the Eso-Twist [69] and reducing the information for defining the grid, such as for the sub-grid-distances file (Listing 3.9), the GPGPU memory consumption is reduced. However, by separating the grid generation from the simulation some advantages are missing, such as the unification of the entire CFD process on the GPGPU [95, 96], the possible implementation of moving geometries [97, 98], the Fluid Structure Interaction (FSI), and the Adaptive Mesh Refinement (AMR) [99, 100]. Nevertheless, using the GPGPU memory

only for the LBM simulation allows for the allocation of very large systems on relative small devices. For giving an example, it was possible to allocate more than 30×10^6 points on a single NVIDIA GeForce GTX TITAN with 6 *Gb* memory and more than 60×10^6 nodes on a single NVIDIA Tesla K40c with 12 *Gb* memory.

4 | Relaxation parameters

We do not need magic to transform our world.
We carry all of the power we need inside ourselves already.
J. K. Rowling

For centuries, the human being tried to give explanations of the natural phenomena experienced. Due to the lack of scientific knowledge, the first approach adopted for finding an explanation was to associate the unknown phenomena to some extra-natural powers. Already from the ancient Greece or before, people established deities causing all the unknowns phenomena. For example, Zeus was responsible for thunders, Poseidon for earthquakes, Aeolus for winds. Hence, every natural phenomenon was surrounded by a magical atmosphere.

In the everyday use of the language, magic can also mean that something posses a quality that is hard to believe, such that people will question whether it is actually true. Similar definition appeared in the lattice Boltzmann method when it was discovered that the numerical error in certain situations could disappear entirely if using "magic" combinations of relaxation rates [46, 47].

In this chapter the effects of the relaxation parameters are examined for various flow test cases.

4.1 Introduction to the Ginzburg's idea

Using a model with two relaxation rates, one for all even moments and one for all odd moments, Ginzburg proved that the relaxation rate for the odd moments has an influence on the accuracy of the results for bounded laminar flows [46]. She showed that by choosing an appropriate combination of the odd and even rates it was possible to obtain the correct solution for the Poiseuille and Couette flow test cases. The relation stated:

$$\Lambda = \frac{4}{3}\left(\frac{1}{\omega_{even}} - \frac{1}{2}\right)\left(\frac{1}{\omega_{odd}} - \frac{1}{2}\right), \tag{4.1}$$

where Λ is the Ginzburg coefficient, ω_{even} is the relaxation parameter directly related to the kinematic viscosity ν (Eq. (2.8) where ω_{even}=ω), and ω_{odd} is the relaxation pa-

rameter for the odd moments. For the magic value $\Lambda = 1/4$ the error in the Poiseuille flow vanished. She also showed that, by keeping the coefficient Λ as constant, the permeability of porous media was independent of the viscosity even for arbitrary complex flows [47]. However, the magic value $\Lambda = 1/4$ was optimal only for the special case of Poiseuille flow. The optimal choice of Λ depends on the particular flow simulated and the position and inclination of the boundary wall relatively to the lattice location. Indeed, the "magic" parameters [101, 102] are good for keeping the error constant but they are not good for eliminating the error, due to the sensitivity to the particular case.

By doing some algebra from Eq. (4.1) it is possible to write the relaxation parameter ω_{odd} as function of ω_{even}:

$$\omega_{odd} = \frac{8(2 - \omega_{even})}{8 - \omega_{even}}. \tag{4.2}$$

An advantage of the cumulant LBM over the two relaxation rates method is that it has more parameters to tune. Instead of having just one odd rate, the cumulant LBM has four independent relaxation rates for the odd cumulants. One of them relates to fifth order cumulants and its influence can be assumed to be small. The remaining three are related to third order cumulants and their act like the Ginzburg's ω_{odd} but with two more degrees of freedom.

4.2 New set of relaxation parameters

The derivation of the new relaxation parameters was not done in this work. Therefore, in this section, I will only explain their purpose without deriving them.

By asymptotic analysis Geier showed that the Navier-Stokes equations arising from the cumulant LBM has five independent linear error terms of leading order (Linearized Leading Error – LLE). For example the NS equation in X-direction has the LLE_x [103]:

$$\begin{aligned} LLE_x = {} & g_1(\omega_1, \omega_2, \omega_3, \omega_4, \omega_5)\partial_{xxxx}u \\ & + g_2(\omega_1, \omega_2, \omega_3, \omega_4, \omega_5)[\partial_{yyyy}u + \partial_{zzzz}u] \\ & + g_3(\omega_1, \omega_2, \omega_3, \omega_4, \omega_5)[\partial_{xxyy}u + \partial_{xxzz}u] \\ & + g_4(\omega_1, \omega_2, \omega_3, \omega_4, \omega_5)[\partial_{xyzz}v + \partial_{xyyz}w] \\ & + g_5(\omega_1, \omega_2, \omega_3, \omega_4, \omega_5)\partial_{yyzz}u. \end{aligned} \tag{4.3}$$

The terms $g_1 \cdots g_5$ are functions of the relaxation rates. The rates ω_1 and ω_2 govern the shear and the bulk viscosity, respectively, and they are fixed by the simulation case. The relaxation rates of the third order cumulants ω_3, ω_4, and ω_5 relate to the

cumulants by the collision operator:

$$\begin{aligned}
C^*_{120} + C^*_{102} &= (1-\omega_3)(C_{120} + C_{102}),\\
C^*_{210} + C^*_{012} &= (1-\omega_3)(C_{210} + C_{012}),\\
C^*_{201} + C^*_{021} &= (1-\omega_3)(C_{201} + C_{021}),\\
C^*_{120} - C^*_{102} &= (1-\omega_4)(C_{120} - C_{102}),\\
C^*_{210} - C^*_{012} &= (1-\omega_4)(C_{210} - C_{012}),\\
C^*_{201} - C^*_{021} &= (1-\omega_4)(C_{201} - C_{021}),\\
C^*_{111} &= (1-\omega_5)C_{111}.
\end{aligned} \tag{4.4}$$

In order to eliminate the LLE, the roots with respect to ω_3, ω_4, and ω_5 of the terms $g_1 \cdots g_5$ have to be found. The problem is that the number of variables is smaller than the number of equations. Therefore, it is not possible to have all the roots at the same time.

Since the problem has no solution, a standard choice is to consider the relaxation rates identical and with a unitary value (set "Cum A"):

$$\omega_3 = \omega_4 = \omega_5 = 1. \tag{4.5}$$

In this way the right hand side of Eq. (4.4) vanishes leading to:

$$\begin{aligned}
|C^*_{120}| &= |C^*_{102}|,\\
|C^*_{210}| &= |C^*_{012}|,\\
|C^*_{201}| &= |C^*_{021}|,\\
C^*_{111} &= 0.
\end{aligned} \tag{4.6}$$

However, this approach ignores the LLE completely. It has been demonstrated that this set of parameters gives accurate results for flows around a sphere until the limit of the drag crisis ($Re < 200\,000$) [43]. With the choice "Cum A" the terms $g_1 \cdots g_5$ do not vanish for viscosity $\omega_1 \to 0$. The remaining error depends on ω_2 and for small viscosity it dominates the physical dissipation.

Although it is not possible to set the terms $g_1 \cdots g_5$ to zero with just three parameters, Geier proposed a combination of the relaxation parameters that eliminates the constant part of the error [104]. In this way, the LLE reduces for smaller viscosity and it does not dominate the physical dissipation. This new set of relaxation parameters for the third order moments was derived by performing a combination of Taylor expansion [105] and asymptotic analysis [106] of Eq. (2.4) until the fifth order. In terms

of Ginzburg parameters, the coefficients are:

$$\Lambda_3 = 1/12, \quad \Lambda_4 = 1/6, \quad \Lambda_5 = 7/24, \tag{4.7}$$

and the new relaxation parameters written as function of ω_1 are (set "Cum B"):

$$\omega_3 = \frac{3(\omega_1 - 2)}{\omega_1 - 3}, \quad \omega_4 = \frac{6(\omega_1 - 2)}{\omega_1 - 6}, \quad \omega_5 = \frac{12(2 - \omega_1)}{12 + \omega_1}. \tag{4.8}$$

Reducing the numerical dissipation leads to a more realistic turbulence intensity. However, it also reduces the stability of the simulation at high Reynolds number. In order to have accurate and stable simulations, Geier proposed to apply a limiter coefficient $c_{lim} \in \mathbb{R}^+$ for the relaxation parameters. The bounded version of the relaxation parameters with c_{lim} becomes (set "Cum B-lim"):

$$\begin{aligned} \omega_{3a}^{lim} &= \omega_3 + \frac{(1-\omega_3)|C_{120} + C_{102}|}{|C_{120} + C_{102}| + \frac{c_{lim}}{\rho}}, \quad \omega_{3b}^{lim} = \omega_3 + \frac{(1-\omega_3)|C_{210} + C_{012}|}{|C_{210} + C_{012}| + \frac{c_{lim}}{\rho}}, \\ &\qquad\qquad \omega_{3c}^{lim} = \omega_3 + \frac{(1-\omega_3)|C_{201} + C_{021}|}{|C_{201} + C_{021}| + \frac{c_{lim}}{\rho}}, \end{aligned} \tag{4.9}$$

$$\begin{aligned} \omega_{4a}^{lim} &= \omega_4 + \frac{(1-\omega_4)|C_{120} - C_{102}|}{|C_{120} - C_{102}| + \frac{c_{lim}}{\rho}}, \quad \omega_{4b}^{lim} = \omega_4 + \frac{(1-\omega_4)|C_{210} - C_{012}|}{|C_{210} - C_{012}| + \frac{c_{lim}}{\rho}}, \\ &\qquad\qquad \omega_{4c}^{lim} = \omega_4 + \frac{(1-\omega_4)|C_{201} - C_{021}|}{|C_{201} - C_{021}| + \frac{c_{lim}}{\rho}}, \end{aligned} \tag{4.10}$$

$$\omega_5^{lim} = \omega_5 + \frac{(1-\omega_5)|C_{111}|}{|C_{111}| + \frac{c_{lim}}{\rho}}, \tag{4.11}$$

where *lim* stands for bounded version with limiter coefficient. The collision of the third order cumulants becomes:

$$\begin{aligned} C_{120}^* + C_{102}^* &= (1 - \omega_{3a}^{lim})(C_{120} + C_{102}), \\ C_{210}^* + C_{012}^* &= (1 - \omega_{3b}^{lim})(C_{210} + C_{012}), \\ C_{201}^* + C_{021}^* &= (1 - \omega_{3c}^{lim})(C_{201} + C_{021}), \\ C_{120}^* - C_{102}^* &= (1 - \omega_{4a}^{lim})(C_{120} - C_{102}), \\ C_{210}^* - C_{012}^* &= (1 - \omega_{4b}^{lim})(C_{210} - C_{012}), \\ C_{201}^* - C_{021}^* &= (1 - \omega_{4c}^{lim})(C_{201} - C_{021}), \\ C_{111}^* &= (1 - \omega_5^{lim})C_{111}. \end{aligned} \tag{4.12}$$

The limiter coefficient acts as follows:

set name	ω_3	ω_4	ω_5
Cum A	1	1	1
Cum B	$\frac{3(\omega_1-2)}{\omega_1-3}$	$\frac{6(\omega_1-2)}{\omega_1-6}$	$\frac{12(2-\omega_1)}{12+\omega_1}$
Cum B-lim	Eq. (4.9)	Eq. (4.10)	Eq. (4.11)

Table 4.1: Different sets of the third order moments relaxation parameters.

- $\omega_{(\cdot)}^{lim} \rightarrow 1$ for $\rho|C_{(\cdot)}| \gg c_{lim}$, it restores the values of the parameters to 1 ("Cum A"),
- $\omega_{(\cdot)}^{lim} \rightarrow \omega_{(\cdot)}$ for $\rho|C_{(\cdot)}| \ll c_{lim}$, it sets the values of the parameters to those ones calculated by Eq. (4.8).

For smooth solution the third order cumulants $C_{(\cdot)}$ do not deviate far from zero and the cumulant LBM uses the optimized relaxation parameters ("Cum B"). However, the third order cumulants may become large when the solution starts to oscillate, leading to unstable solutions. In this case, the c_{lim} acts to set the relaxation parameters to the stable values ("Cum A").

Table 4.1 gives a summary of the different sets for the relaxation parameters of third order cumulants used in this work.

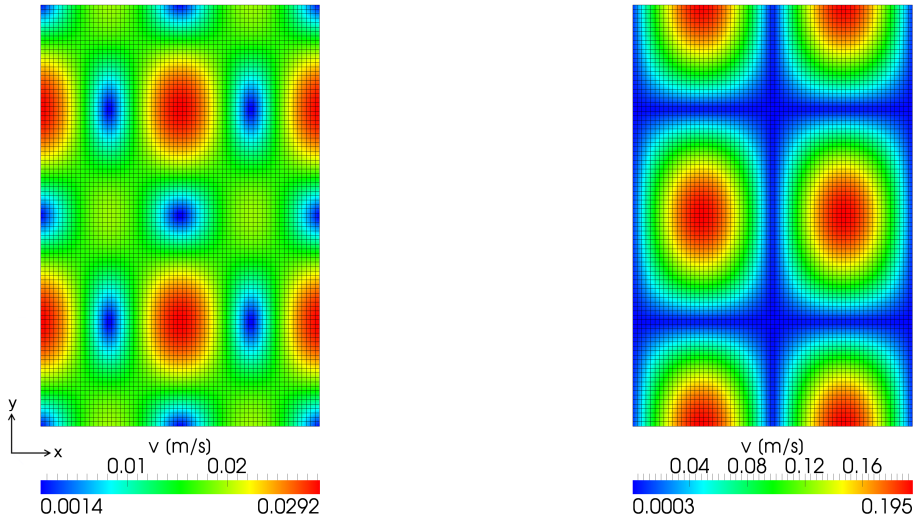
4.3 Results

In this section various validation test cases for the different sets of relaxation rates of the third order cumulants were conducted. The cases were the Taylor Green vortex, the shear wave, and the Poiseuille flow for different channel inclinations. Since the cases were laminar flows, only the first two sets "Cum A" and "Cum B" were used. For turbulent flows at high Re the new parameters "Cum B" have to be bounded and the set "Cum B-lim" was necessary for having stable simulations, especially for coarse grids (Chapter 5).

The grids for the benchmarks were generated by using `LBMHexMesh` (Chapter 3) and the simulations carried out with `LBMCumulantFoam` (Appendix B).

4.3.1 Taylor Green vortex test case

The first test case was the Taylor Green vortex [107]. The domain was $L \times 1.5L \times 3\Delta x$ with $L = 5.12\ m$, and all the boundary faces were periodic condition. The flow was initialized by providing a velocity flow field $\vec{u}_{init} = (u_{init}, v_{init}, w_{init})$ as function of



(a) Taylor Green vortex

(b) Shear wave

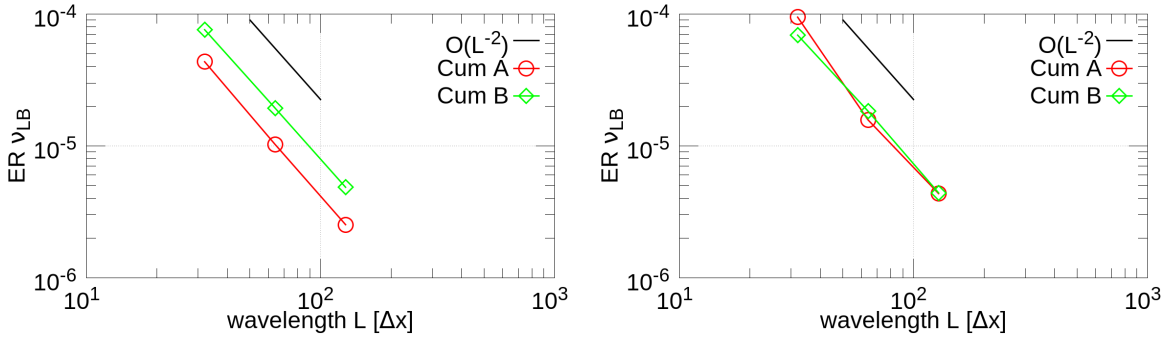
Figure 4.1: Taylor Green vortex and Shear wave. Taylor Green vortex (a) and shear wave (b) test cases, medium resolution ($L64$), lattice viscosity $\nu_{LB} = 10^{-2}\Delta x^2/\Delta t$.

cosine and sine of the node location for the two directions along the plane (Figure 4.1a):

$$\begin{aligned} u_{init} &= u_0/L \sin(2\pi x/L)\cos(4\pi y/(3L)), \\ v_{init} &= -3.0u_0/2/L \cos(2\pi x/L)\sin(4\pi y/(3L)), \\ w_{init} &= 0, \end{aligned} \tag{4.13}$$

where $u_0 = 0.1\ m/s$ was the amplitude of the velocity. In order to perform a convergence study, three grids with different resolutions were generated with $\Delta x = L/32$, $L/64$, and $L/128$. Thus, the grid refinement ratio r, the ratio between two successive grid resolutions, had the value of 2. By changing the grid resolution, in order to keep the Reynolds number constant, diffusive scaling with the time-step $\Delta t \propto \Delta x^2$ was used. In addition, two different viscosities $\nu_{LB} = 10^{-2}\Delta x^2/\Delta t$ and $10^{-3}\Delta x^2/\Delta t$ in lattice units were tested.

The rate of decay of the vortex was measured and the error in the recomputed viscosity was calculated. Figure 4.2 shows the normalized error in viscosity for the three grid resolutions and the two lattice viscosities by using the two different sets of the relaxation parameters. With both the sets the simulations were second order accurate. The new set "Cum B" showed a larger error than the set "Cum A" for the large viscosity (Figure 4.2a). Unlike the results for the set "Cum A", the error for the set "Cum B" did not increase for smaller viscosity (Figure 4.2b), which was the aim in the derivation of the new relaxation parameters.



(a) $\nu_{LB} = 10^{-2}\Delta x^2/\Delta t$ (b) $\nu_{LB} = 10^{-3}\Delta x^2/\Delta t$

Figure 4.2: Taylor Green vortex – convergence study. Error in viscosity by measuring the rate of decay of a Taylor Green vortex for different grid resolutions and viscosities.

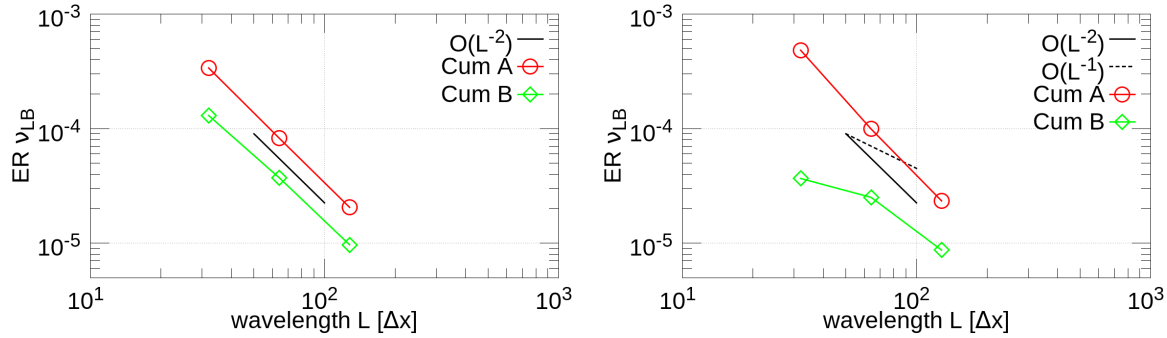
4.3.2 Shear wave test case

The second test case was the shear wave flow. Similarly to the previous case, the domain was $L \times 1.5L \times 3\Delta x$ with $L = 5.12\ m$, and all the boundary faces were periodic condition. The flow was initialized by providing a velocity $\vec{u}_{init} = (u_{init}, v_{init}, w_{init})$ as function of cosine and sine of the node location for the direction normal to the plane (Figure 4.1b):

$$\begin{aligned} u_{init} &= 0, \\ v_{init} &= 0, \\ w_{init} &= 1/L \sin(2\pi x/L) \cos(4\pi y/(3L)). \end{aligned} \tag{4.14}$$

Grid resolutions and settings were the same as the Taylor Green vortex benchmark: $\Delta x = L/32$, $L/64$, and $L/128$ ($r = 2$), the time-step was $\Delta t \propto \Delta x^2$, and two different viscosities $\nu_{LB} = 10^{-2}\Delta x^2/\Delta t$ and $10^{-3}\Delta x^2/\Delta t$ in lattice units were tested.

As for the Taylor Green vortex case, the rate of decay of the wave was measured and the error in the recalculated viscosity was computed. Figure 4.3 shows the normalized error in viscosity for the three grid resolutions and the two viscosities by using the two different sets of the relaxation parameters. With both the sets the simulations for the large viscosity were second order accurate. The new set "Cum B" gave a smaller error than the set "Cum A" (Figure 4.3a). On the contrary, for the small viscosity the new set "Cum B", even though having a smaller error than the set "Cum A", it was not possible to determine the order of accuracy since the curve had no constant steepness in the log-log plot. Using the set "Cum A", the error increased slightly by decreasing the viscosity. With the set "Cum B", the error decreased with lower viscosity.

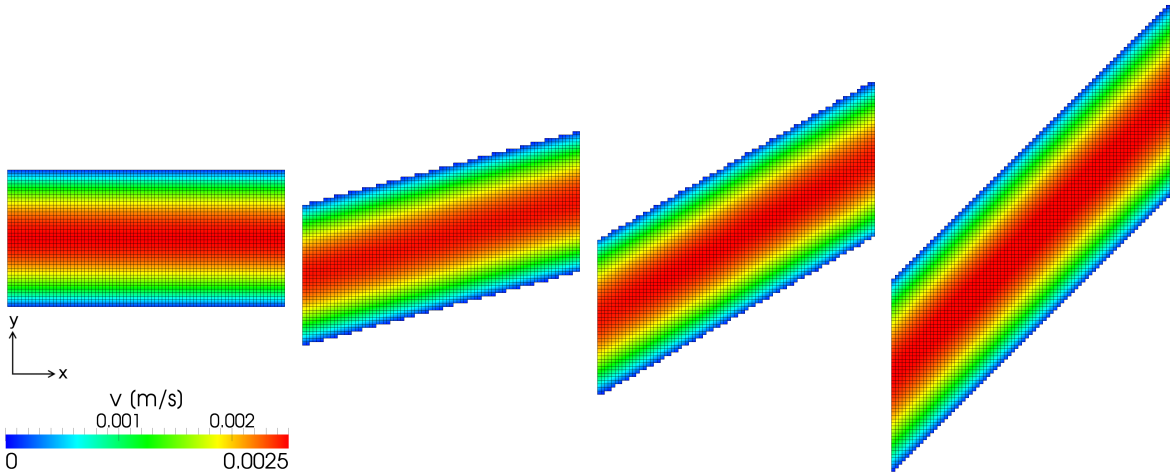


(a) $\nu_{LB} = 10^{-2}\Delta x^2/\Delta t$ (b) $\nu_{LB} = 10^{-3}\Delta x^2/\Delta t$

Figure 4.3: Shear wave – convergence study. Error in viscosity by measuring the rate of decay of a shear wave for different grid resolutions and viscosities.

4.3.3 Poiseuille flow test case

The last benchmark was about the flow between two infinite planes (Poiseuille flow). Four different configurations of the planes were generated with an inclination of 0°, 15°, 30°, and 45° (Figure 4.4). For the case with 0° inclination, the domain was $2N \times N \times 2\Delta x$ with $N = 2\ m$. For the other angle configurations the vertical and horizontal directions of the background mesh were elongated in order to cover all the channel volume with the same channel length of the 0° inclination case. The flow was driven by a constant acceleration of $5 \times 10^{-5}\ m/s^2$ parallel to the planes. The planes were walls with interpolated bounce-back (ibb) no-slip condition, and all the



(a) 0° (b) 15° (c) 30° (d) 45°

Figure 4.4: Poiseuille flow. Poiseuille flow test case for different channel inclinations, medium resolution ($N40$), lattice viscosity $\nu_{LB} = 10^{-2}\Delta x^2/\Delta t$.

other lateral planes were periodic conditions. In order to perform a convergence study, three grids for each angle configurations were generated with different resolutions, with $\Delta x = N/20$, $N/40$, and $N/80$ points ($r = 2$). Diffusive scaling with the time-step $\Delta t \propto \Delta x^2$ was used in order to keep the Reynolds number constant, and two values of viscosity $\nu_{LB} = 10^{-1}\Delta x^2/\Delta t$ and $10^{-2}\Delta x^2/\Delta t$ in lattice units were tested.

After reaching the steady state solution, the velocity profile at the centre of the channel along the normal of the planes was checked, and compared to the analytical solution. For the 0° angle configuration, the sample of points used for plotting the numerical results was located along the normal of the channel planes. Due to the Cartesian grids, the sample of points for the inclined cases were not located along the normal direction of the walls at the centre of the channel (Figure 4.5). They were always aligned to the grid along the vertical direction (y). Therefore, in order to have the correct distance normal to the wall $y_{w,o}$, the node locations were translated by:

$$y_{w,o} = (y - x\tan(\theta))\cos(\theta), \tag{4.15}$$

where x and y are the coordinates of the node, and θ is the angle of the channel inclination.

Figures 4.6a-4.6d and 4.7a-4.7d show the difference between the numerical velocity and the analytical velocity for each point of the profile. The results are shown for the two different viscosities by simulating with the set of the relaxation parameters "Cum

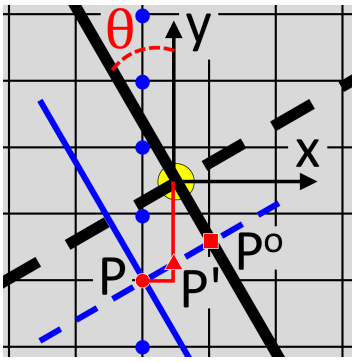
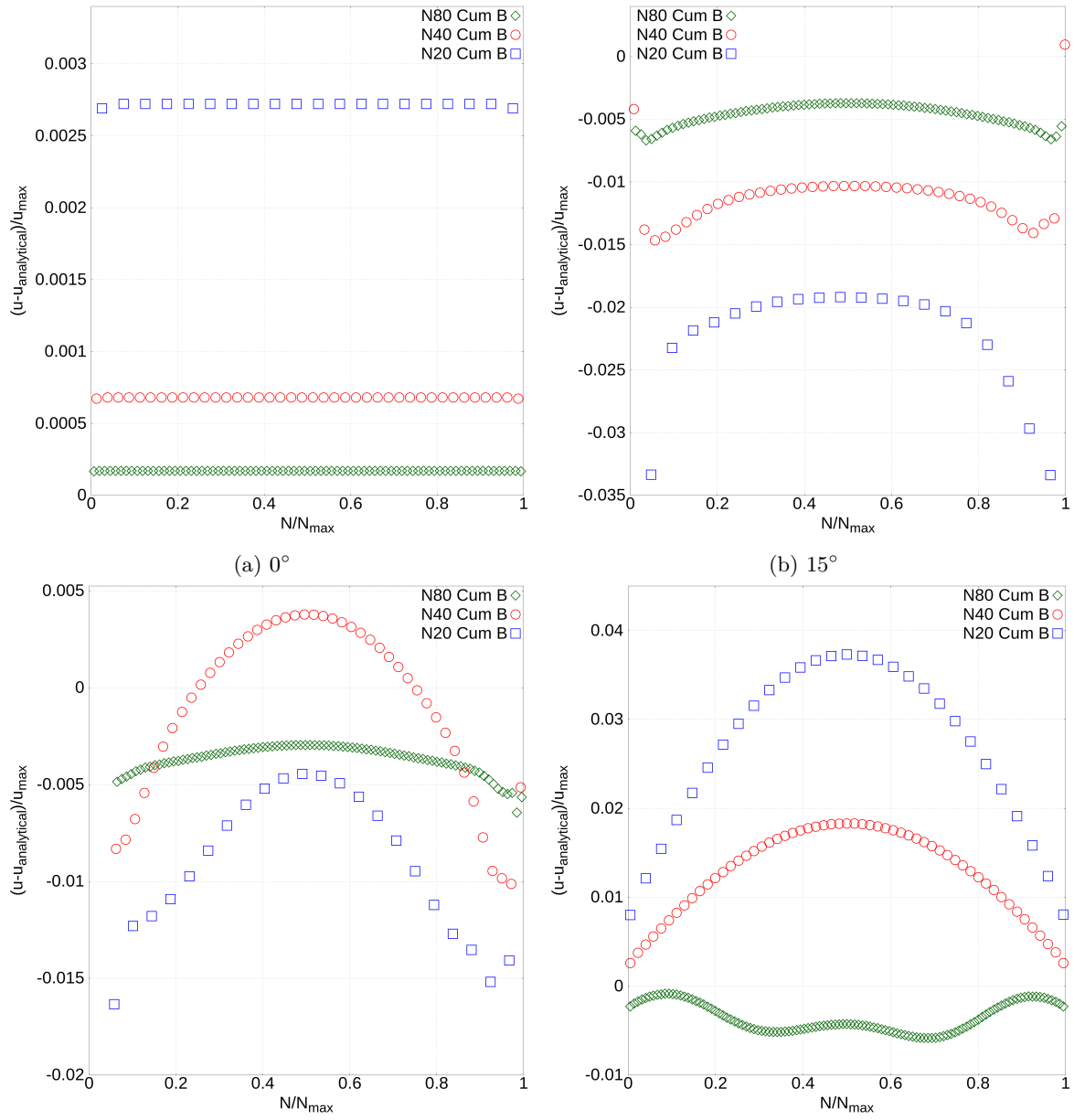


Figure 4.5: Poiseuille flow – reconstruction of the normal distance to the wall. The normal to the wall (black thick line) had an angle θ with respect to the vertical direction (y). The big circle at the centre is the origin (0,0,0) and the dashed thick line is the channel centre-line. Due to the Cartesian grid, the sample of nodes were taken always along y (circles). In order to have the correct position normal to the wall at the channel centre, each node (P) was translated first along the parallel to the channel centre-line (triangle, P') and finally along the normal direction to the wall at the channel centre (square, P^o).



(a) 0°

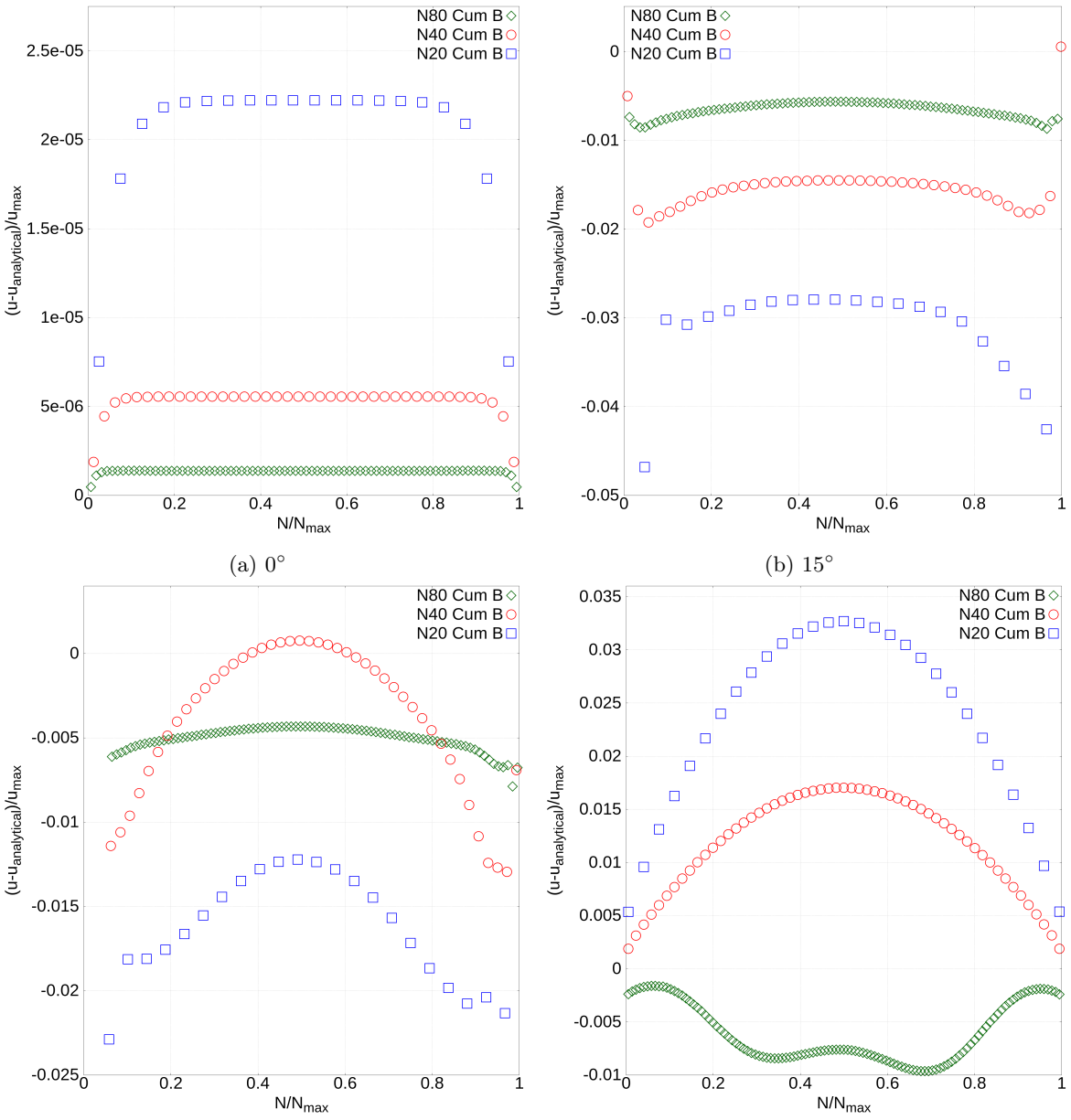
(b) 15°

(c) 30°

(d) 45°

Figure 4.6: Poiseuille flow – difference of the velocity profiles ($\nu_{LB} = 10^{-1}\Delta x^2/\Delta t$). Difference of the velocity profiles for the Poiseuille flow test case for different channel angle configurations, lattice viscosity $\nu_{LB} = 10^{-1}\Delta x^2/\Delta t$, set of the relaxation parameters "Cum B", and different grid resolutions.

B". The curves are plotted by using Eq. (4.15) for the distance to the wall, while the numerical velocity was the magnitude of the two components in X- and Y-direction. For all the profiles, the distance and the velocity were normalized to the channel height (N/N_{max}) and the maximum analytical velocity ($(u - u_{analytical})/u_{max}$), respectively.

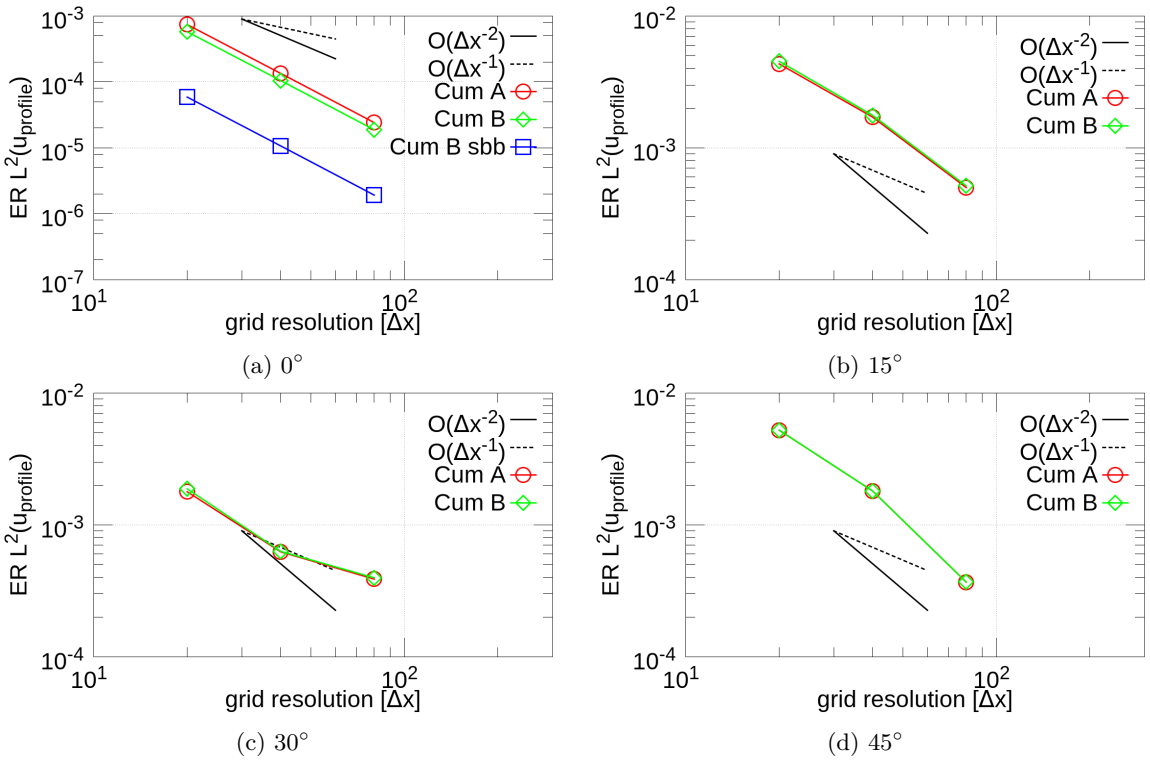


(a) 0° (b) 15°

(c) 30° (d) 45°

Figure 4.7: Poiseuille flow – difference of the velocity profiles ($\nu_{LB} = 10^{-2}\Delta x^2/\Delta t$). Difference of the velocity profiles for the Poiseuille flow test case for different channel angle configurations, lattice viscosity $\nu_{LB} = 10^{-2}\Delta x^2/\Delta t$, set of the relaxation parameters "Cum B", and different grid resolutions.

The numerical results for the 0° angle configuration gave a good agreement with the analytical solution for both the viscosities. The relative error was small and it showed a grid convergence. For all the inclined cases the relative error changed its sign while refining the grid. Interestingly, the numerical results were not symmetric for the 15°



(a) 0° (b) 15°

(c) 30° (d) 45°

Figure 4.8: Poiseuille flow – convergence study ($\nu_{LB} = 10^{-1}\Delta x^2/\Delta t$). Error by measuring the L^2 norm of the velocity profile for different grid resolutions, lattice viscosity $\nu_{LB} = 10^{-1}\Delta x^2/\Delta t$.

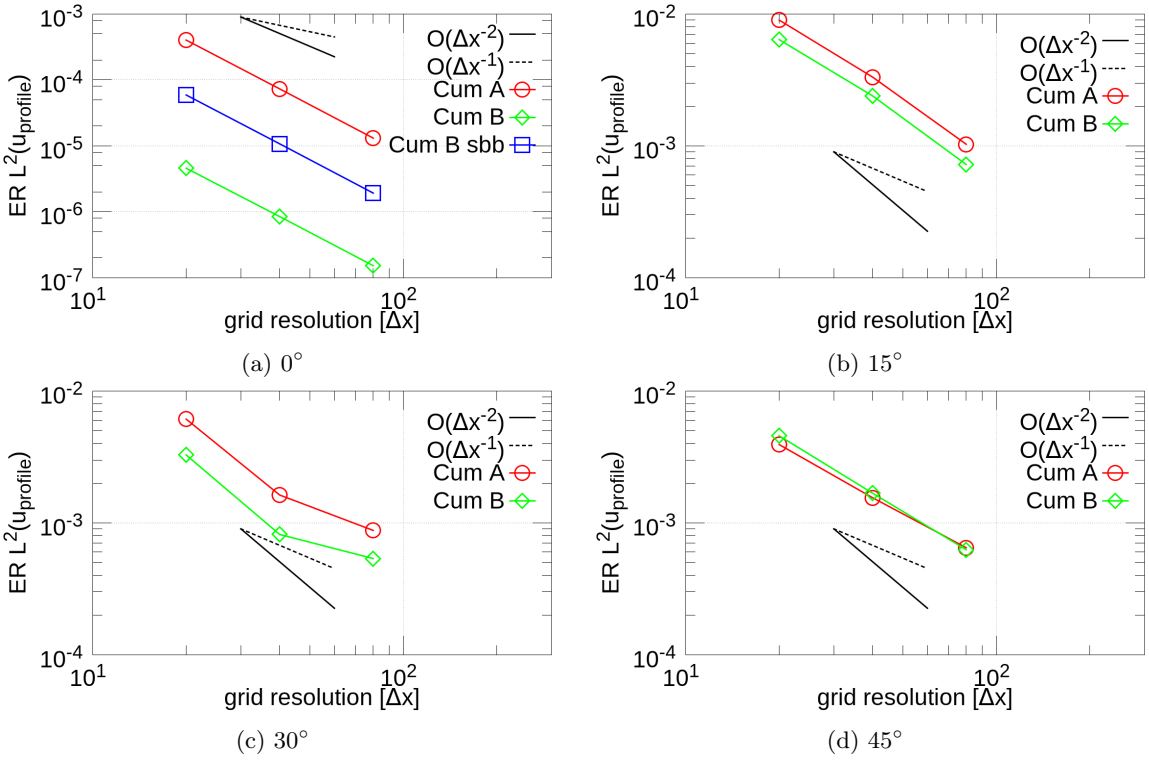
and 30° angle configurations due to the different distance of the nodes from the bottom and from the top planes. The symmetry was recovered by the 45° angle case because the grid was symmetric with respect to the channel centre-line, like for the 0° configuration.

Figures 4.8 and 4.9 report the normalized error in velocity for the sample of nodes translated to the centre of the channel along the normal direction of the planes. The error was computed by using the least square method (L^2) in order to have a single indicator for the deviation of the complete velocity profile from the analytical solution:

$$L^2 = \frac{1}{n_p}\sqrt{\sum_{n=1}^{n_p}\left(\frac{u_n - u_a}{u_{max}}\right)^2}, \tag{4.16}$$

where n_p is the number of sample nodes, u_n is the numerical velocity at the n-th node, u_a is the analytical velocity at the same position of the n-th node, and u_{max} is the maximum analytical velocity.

For the 0° angle configuration the ibb boundary condition showed a second order accuracy, and the new set of parameters "Cum B" showed a smaller error than the



(a) 0° (b) 15°

(c) 30° (d) 45°

Figure 4.9: Poiseuille flow – convergence study ($\nu_{LB} = 10^{-2}\Delta x^2/\Delta t$). Error by measuring the L^2 norm of the velocity profile for different grid resolutions, lattice viscosity $\nu_{LB} = 10^{-2}\Delta x^2/\Delta t$.

set "Cum A", especially for the lower viscosity $\nu_{LB} = 10^{-2}\Delta x^2/\Delta t$ (Figures 4.8a and 4.9a). However, the set "Cum B" showed an error dependence on the viscosity. The errors were smaller for the lower viscosity than for the higher one. The reason could be the use of the ibb boundary condition. For the 0° angle configuration with the planes located at the half grid spacing ($q = 0.5$), the ibb differed from the sbb. This is shown in Eq. (2.18) *versus* Eq. (2.14). Hence, the case 0° for the set "Cum B" was simulated with the sbb boundary condition. As stated by Ginzburg [47], the results showed error independence of the viscosity ("Cum B sbb" in Figures 4.8a and 4.9a).

With a low channel angle of 15° the convergence remained of second order. With respect to the set "Cum A", the new set of parameters "Cum B" showed similar errors for the higher viscosity, and lower errors for the lower viscosity (Figures 4.8b and 4.9b).

With 30° inclination angle the convergence reduced globally to first order accuracy, and the new set "Cum B" showed lower errors than "Cum A" only for the lower viscosity case (Figures 4.8c and 4.9c).

Finally, the simulations for the channel configuration with 45° inclination angle

showed a quasi second order accuracy for the higher viscosity $\nu_{LB} = 10^{-1}\Delta x^2/\Delta t$ and a quasi-second order accuracy for the lower viscosity $\nu_{LB} = 10^{-2}\Delta x^2/\Delta t$ (Figures 4.8d and 4.9d). Both sets "Cum B" and "Cum A" showed similar errors.

4.4 Discussion and summary

The new set of the relaxation parameters "Cum B" showed a general second order convergence accuracy for both not-bounded (Taylor Green vortex, shear wave) and bounded test cases (Poiseuille flow). For the latter case, in combination with the interpolated bounce-back boundary condition, "Cum B" showed mixed results for straight and inclined walls. For example, in the 30° and 45° angle cases it was not possible to determinate exactly the order of accuracy. The error in velocity changed the sign while refining the grid: the velocity profile was above or below the analytical solution depending on the grid resolution (Figures 4.6c, 4.6d, 4.7c, and 4.7d). This implies that the error was not yet asymptotic. Therefore, higher resolutions are required for measuring the true order of convergence.

As predicted by Ginzburg [47], the error was independent of the viscosity when "magic" parameters were used (Λ being constant). This was true for the Taylor Green vortex test case and in part for the shear wave test case. For the Poiseuille flow test case with the new set "Cum B", the error in the velocity profile was dependent on the viscosity, already for straight planes with 0° angle. The reason for this discrepancy was the interpolated bounce-back (ibb) boundary condition for straight walls defined in Eq. (2.18). Nevertheless, for the small viscosity the error using the ibb was actually smaller than the error for the sbb. The set "Cum B", although showing dependence of the error on the viscosity for bounded flows, gave smaller errors than the set "Cum A".

The new set of relaxation parameters "Cum B" was derived for having the terms $g_1 \cdots g_2$ of the Linearized Leading Error not depending on the bulk viscosity (ω_2). However, the solution of "Cum B" is not unique for this constraint. Another choice could yield better results.

5 | Wall function

Dichotomy paradox. That which is in locomotion must arrive at the half-way stage before it arrives at the goal.
Aristotle, *Physics*

The dichotomy paradox plays with the distance to reach the end of a path. Before anyone could cover a predefined distance, he must cover a half of it. Before travel the half way he must cover a quarter, and so on, becoming the distance infinitesimally small. Hence, before he could reach the end of the path he must walk infinite number of steps.

In Computational Fluid Dynamics the discretization of the fluid domain in the near-wall region deals with a similar problem. The closer to the wall the higher resolution is required to resolve the velocity gradients. While it is not necessary to resolve the infinitesimally small, the smallest scales in the flow are very small for large enough Reynolds number. The advances in computational capacity are not sufficient for solving the problem of very large systems. To overcome this limitation, the key is the near-wall region treatment with the use of wall functions.

In this chapter the near-wall treatment for turbulent flows at high Reynolds numbers with the cumulant LBM is investigated, leading to the development of a new wall function for modelling the turbulent boundary layer (TBL). Before explaining the new boundary condition, the TBL theory is introduced. Finally, numerical results of turbulent bounded flow simulations are shown.

5.1 Introduction to the turbulent boundary layer theory

The turbulent boundary layer concept was first described by Ludwig Prandtl in 1904 [1, 108]. He hypothesized that the boundary layer is a fluid flow region near the body surface where the effects of viscosity are predominant [109, 110]. One of the major effects is the friction between the fluid and the body, which causes that the fluid immediately adjacent to the surface sticks to the wall (no-slip condition). Moreover, since the velocity at the wall is zero while outside the boundary layer it is relatively high, the boundary layer is a region of large velocity gradients. According to the

Newton's shear-stress law:

$$\tau_w = \mu \partial_n u_e, \tag{5.1}$$

which states that the wall shear stress τ_w is proportional to the velocity gradient $\partial_n u_e$, the local wall shear stress in the boundary layer can be very large (u_e is in the stream-wise direction $\vec{e}$, ∂_n is in the direction normal to the wall $\vec{n}$, and μ is the dynamic viscosity). Thus, the skin-friction drag force acting on the body is not negligible, contrary to what in the earlier 19$^{\text{th}}$ century scientist believed [110]. It is possible to define the skin-friction coefficient C_f as:

$$C_f = \frac{\tau_w}{1/2 \rho u_\infty^2}, \tag{5.2}$$

with u_∞ being the free-stream velocity. Several approximation formulas for C_f are available in literature for both laminar [111] and turbulent flows [109].

Another effect of the friction at the wall is the flow separation. Prandtl theorized that the fluid inside the boundary layer, having dissipated a part of its kinetic energy due to the friction, does not have sufficient energy to move uphill into the region where the pressure is higher, and therefore it turns away from the boundary (Figure 5.1) [110]. Hence, the velocity profile near the surface is depleted, the flow separates from the wall, and the boundary layer thickness δ increases (velocity profile on the right of Figure 5.1). Once the flow separates, the pressure distribution over the surface of the body changes drastically. For large flow separation regions the pressure drag is usually greater than the skin-friction drag.

The boundary layer thickness δ is related to the Reynolds number Re. The higher the Reynolds number the thinner is the boundary layer thickness [109]. For turbulent

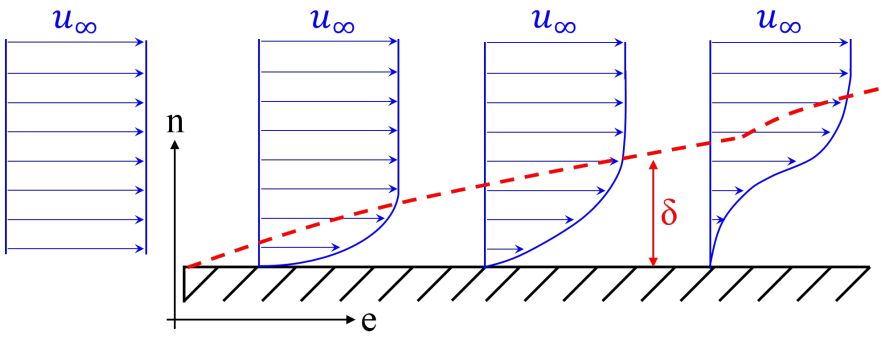


Figure 5.1: Sketch of the boundary layer. The boundary layer is the fluid region between the wall and the free-stream velocity u_∞ (below the dashed line). δ is the boundary layer thickness, $\vec{n}$ is the normal to the wall, and $\vec{e}$ is the stream-wise direction.

flows at high Re the boundary layer thickness becomes extremely smaller than the reference length of the geometry ($\delta \ll L$). Hence, in order to solve numerically the boundary layer, it is necessary to provide a highly refined grid close to the wall with at least [112]:

$$n_e \approx 10, \quad n_n \approx 25, \quad n_s \approx 10, \tag{5.3}$$

where n_e, n_n, and n_s are the numbers of points stream-wise, across, and span-wise the boundary layer thickness, respectively.

5.1.1 Thin boundary layer approximation

The study of near-wall region flows leads to the development of the so-called wall functions [32–41]. They are used in order to avoid very large meshes for solving the turbulent boundary layer over complex geometries at high Re. By using wall functions the boundary layer effects are modelled and the boundary layer thickness δ does not have to be resolved with a very fine grid resolution.

One possible method for implementing a wall function is the thin boundary layer approximation [109, 113, 114]. It allows to simplify the NS momentum transport equation by the following hypotheses: i) neglect the diffusion processes parallel to the body surface, and ii) replace the momentum equation normal to the surface with the assumption of zero normal pressure gradient throughout the boundary layer. The mean flow is assumed to be parallel to the wall and statically steady:

$$u_n \ll u_e, \quad \partial_e u_e \ll \partial_n u_e, \quad \partial_t u_e = 0, \quad \partial_n p = 0, \tag{5.4}$$

where u_e and u_n are the velocity components, and ∂_e and ∂_n are the derivatives in the stream-wise direction $\vec{e}$ and in the direction normal to the wall $\vec{n}$, respectively (∂_t is the time derivative). In the TBL Eq. (1.1) reduces to:

$$0 = -\partial_e p + \partial_n(\mu \partial_n u_e) + \rho g_e, \tag{5.5}$$

where the term $-\partial_e p$ is the pressure gradient in the stream-wise direction $\vec{e}$, and $\mu \partial_n u_e$ is the wall shear stress τ_w of Eq. (5.1).

Eq. (5.5) is a one-dimensional equation for the second derivative of the velocity. A standard approach to solve Eq. (5.5) is to integrate in the direction normal to the wall and solve for the first derivative of the velocity with a Gauss-Legendre quadrature method [115]. The method has to be supplemented by a Newton algorithm for obtaining τ_w. Other approaches consider the von Kármán integral TBL and they use

approximation methods for recovering the information at the wall [109]. Importantly, with Eq. (5.5) the TBL is solved taking into account the pressure gradient effects, which are necessary for separating flows (e.g. at curved walls).

5.1.2 Second derivative of the velocity for flat walls

In the case of flat walls, the pressure gradient term $-\partial_e p$ vanishes and Eq. (5.5) is only a function of the second derivative of the velocity $\partial_n(\mu\partial_n u_e)$ (zero pressure gradients condition). For this reason, the important quantities can be computed by taking into account only the effects of the velocity, leading to quasi-analytical solutions such as the "law of the wall" [29], the Musker law [35], and the Monkewitz law [36].

With the cumulant LBM, after performing asymptotic analysis of Eq. (2.4) until the third order in diffusive scaling, the relationship between the third order cumulants and the second derivative of the velocity is found (Appendix C). For giving an example, considering a flat wall with $\vec{n} = (0, 1, 0) = y$ and $\vec{e} = (1, 0, 0) = x$, the second derivative of the stream-wise velocity $u_e = u$ in the direction normal to the wall $\partial_{nn} = \partial_{yy}$ can be computed from [103]:

$$\frac{C^*_{120} - C_{120} - {}^1\!/_3\rho g_x}{-\frac{2}{9}\rho\left(\frac{1}{\omega_1} - \frac{1}{2}\right)} = 2\partial_{xy}v + \partial_{yy}u + \mathcal{O}(\Delta x^2), \tag{5.6}$$

where ω_1 is the relaxation rate related to the viscosity, and C^*_{120} and C_{120} are the third order cumulants in xyy for the post-collision and pre-collision states, respectively. By considering the thin boundary layer approximation ($v \ll u$), the mixed derivative $\partial_{xy}v$ can be neglected:

$$\partial_{yy}u \approx \frac{C^*_{120} - C_{120} - {}^1\!/_3\rho g_x}{-\frac{2}{9}\rho\left(\frac{1}{\omega_1} - \frac{1}{2}\right)}. \tag{5.7}$$

This allows to solve directly the second derivative of the velocity of Eq. (5.5) and to obtain the wall shear stress τ_w. For the first fluid node close to the wall it is possible to write:

$$\begin{aligned} \partial_y(\mu\partial_y u) = \partial_y(\tau_w) &\approx \Delta\tau_w/\Delta y, \\ \mu\partial_{yy}u &= \frac{\tau_{xy} - \tau_w}{y_w}, \\ \tau_w &= \tau_{xy} - y_w\mu\partial_{yy}u, \end{aligned} \tag{5.8}$$

where y_w is the distance to the wall, and the deviatoric stress tensor component τ_{xy} is

locally evaluated at the first fluid node close to the wall by using cumulants:

$$-C_{110}\frac{3\omega_1}{\rho} = \partial_x v + \partial_y u + \mathcal{O}(\Delta x^2), \tag{5.9}$$

$$\tau_{xy} \approx \mu\left(-C_{110}\frac{3\omega_1}{\rho}\right), \tag{5.10}$$

being C_{110} the second order cumulant in xy.

5.2 Frictional partial slip velocity wall function

The lattice Boltzmann method has only recently been used for the implementation of a near-wall region treatment. In 2014 Malaspinas implemented a wall function which reconstructs the distributions f of the boundary nodes by adding a deviatoric stress term [48]. Malaspinas used the MRT-LBM with a Smagorinsky turbulence model and a Van Driest damping function for the boundary nodes. He validated the wall function by computing the turbulent channel flow test case at different Reynolds numbers by computing both the quasi-analytical solution and the solution of the TBL equation. Malaspinas' wall function considers the fluid density and velocity at the second node from the wall in the normal direction, and recomputes their values at the boundary node. Although the wall function showed good results in terms of velocity and Reynolds shear stress profiles, using two nodes could be a disadvantage. For curved geometries the two nodes are not aligned with the grid and interpolations are required. The use of two nodes with interpolation is not optimal for implementations on GPGPUs. Locality of the information is essential for efficiency of the GPGPU.

This work shows a new wall function for the LBM which uses only information at the boundary nodes. After recomputing the quantities at the wall, the wall function imposes a partial slip velocity at the boundary surface in order to satisfy the skin friction requirement.

5.2.1 Implementation

The frictional partial slip velocity wall function (FPSV-WF) bases on five inputs. They are the fluid velocity $\vec{u}$, the deviatoric stress tensor $\boldsymbol{\tau}$, the second derivative of the velocity $\partial_{nn}u_e$, the normal to the wall $\vec{n}$, and the distance to the wall y_w. The inputs $\vec{u}$, $\boldsymbol{\tau}$, and $\partial_{nn}u_e$ are related to the evolution of the flow and they are computed and updated by the LBM kernel, while $\vec{n}$ and y_w are geometrical information provided by the grid generator `LBMHexMesh` (Chapter 3) and they are constant during the

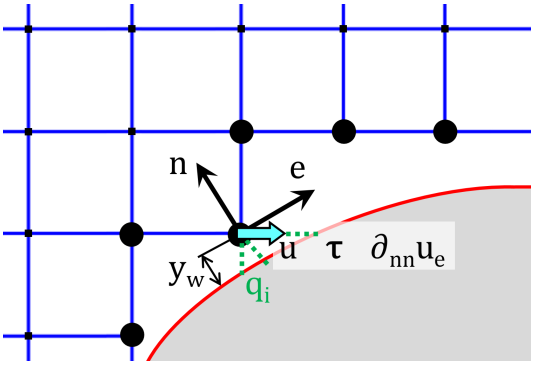


Figure 5.2: Boundary node for the FPSV-WF. The information necessary for the FPSV-WF is local and stored at the first fluid node close to the wall. The sub-grid-distance q is used for the interpolated bounce-back operation.

simulation. All the inputs are localized on the first fluid node next to the wall.

The configuration of the boundary node is shown in Figure 5.2. The first fluid node next to the wall is both boundary node and fluid node. It owns the information from the mesh generation, $\vec{n}$ and y_w, and the LBM kernel computes the standard streaming and collision operations, and $\vec{u}$, $\boldsymbol{\tau}$, and $\partial_{nn}u_e$ can be computed from the cumulants.

The FPSV-WF acts in five steps (Figure 5.3). The first step computes the streamwise direction $\vec{e}$ at the first fluid node by using Eq. (1.19).

The second step calculates the wall shear stress τ_w. Two methods for computing

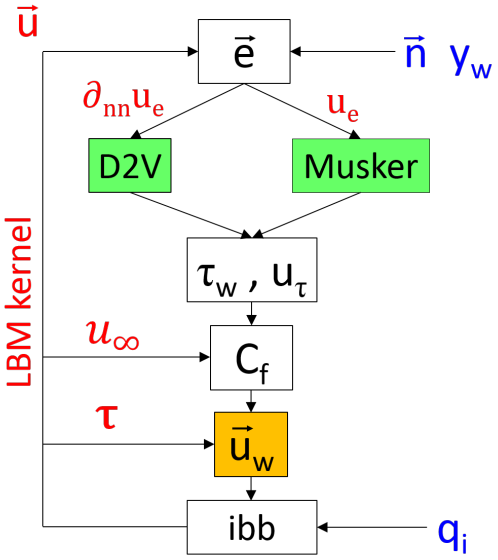


Figure 5.3: Functional flow block diagram of the FPSV-WF.

τ_w are implemented, either by solving the second derivative of the velocity for flat walls with cumulants (D2V) or by solving the quasi-analytical solution from Musker [35]. D2V is solved by using Eq. (5.8). To compute the quasi-analytical solution it is necessary to calculate the stream-wise velocity $u_e = \vec{u} \cdot \vec{e}$ and to solve the Musker law by using a Newton algorithm. After obtaining τ_w, with both methods the frictional velocity u_τ is computed with Eq. (1.20).

Once u_τ is obtained, the skin-friction coefficient C_f can be calculated (third step). By substituting Eq. (1.20) into Eq. (5.2), it is possible to write:

$$C_f = 2\frac{u_\tau^2}{u_\infty^2}. \tag{5.11}$$

For simulations where a force term is used, the free-stream velocity u_∞ is evaluated as the mean velocity in the bulk $\langle u_b \rangle$. The symbol $\langle \cdot \rangle$ indicates the average in space. When setting an inlet boundary condition where the velocity is known, the reference velocity can be used for u_∞.

With the skin-friction coefficient a partial slip velocity can be imposed at the wall $\vec{u}_w$ [40] (fourth step):

$$\vec{u}_w = -\frac{1}{C_f}\vec{n}^T \cdot \boldsymbol{\tau} \cdot \vec{e}, \tag{5.12}$$

where $\vec{n}^T$ is the transpose vector of $\vec{n}$. The deviatoric stress tensor $\boldsymbol{\tau}$ is locally evaluated at the boundary fluid node by using cumulants, e.g. Eqs. (5.9) and (5.10) for τ_{xy} (for the other tensor components the equations are similar by exchanging the indices).

The slip length s can be defined as [116]:

$$s = -\frac{\mu}{C_f}. \tag{5.13}$$

Finally, with $\vec{u}_w$ the wall function performs the fifth step, that is the interpolated bounce-back of Eq. (2.15) taking into account the sub-grid-distance q.

5.3 Results

The FPSV-WF was validated by conducting numerical simulations of the turbulent channel flow test case for different Reynolds numbers, grid resolutions, and methods for computing τ_w.

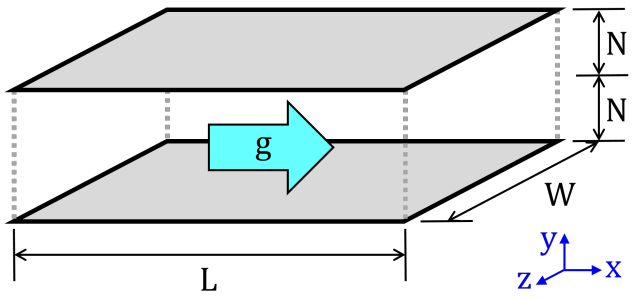


Figure 5.4: Channel flow. Channel flow between two infinite planes, at the bottom and at the top. All the other sides are periodic boundary conditions. The height of the channel is $H = 2N$, with $N = 1\ m$. The length and width are $L = W = 3H$ and the flow is driven by an acceleration $\vec{g}$.

5.3.1 Computational domain

The geometry of the test case is shown in Figure 5.4. The flow has been confined between two infinite planes, one on the bottom and one on the top. All the other sides were periodic boundary conditions. The directions were stream-wise (X), normal (Y), and span-wise (Z). The height of the channel was $H = 2N$ with $N = 1\ m$ being the half channel height. In order to let the turbulence develop in both stream-wise and span-wise directions, the length L and the width W of the channel were set three times the height H ($L = W = 3H$) [48]. Uniform grids were generated by discretizing the half channel height N with three different resolutions: a very coarse one with 10 points, a less coarse one with 20 points, and third one with 40 points. Since the LBM computes on grids with cell aspect ratio equal to one, the discretization in the other two directions (L and W) was of the same resolution. Table 5.1 shows the number of points used for the three different meshes. Importantly, the choice of such coarse grids was intentional in order to stress the FPSV-WF with an under-resolved near-wall region mesh.

The flow moved inside the channel in the stream-wise direction (X) driven by an acceleration $\vec{g}$. The acceleration $\vec{g}$ was set adaptively in order to have a specific space-

grid name	L	W	N	H *[# points]*	Δx *[m]*	Δt *[s]*
$N10$	60	60	10	20	0.1	0.008
$N20$	120	120	20	40	0.05	0.002
$N40$	240	240	40	80	0.025	0.0005

Table 5.1: Summary of the grid information for the turbulent channel flow simulations.

average velocity in the bulk domain $\vec{u}_{b,0}$ [117]:

$$\vec{g} = \left(\frac{\langle u_\tau \rangle^2}{N} + \frac{(u_{b,0} - \langle u_b \rangle) u_{b,0}}{N}, 0, 0 \right), \tag{5.14}$$

where $\langle u_\tau \rangle$ is the average of the computed u_τ over all the boundary nodes, and $\langle u_b \rangle$ the average over all the nodes in the bulk. The specific velocity $\vec{u}_{b,0} = (\nu Re/H, 0, 0)$ was calculated according to the simulated Reynolds number $Re = u_{b,0}H/\nu$. The Re was computed from the frictional Reynolds number $Re_\tau = u_\tau N/\nu$ by using the Dean correlation [48]:

$$Re = \left(\frac{8}{0.073} \right)^{4/7} Re_\tau^{8/7}. \tag{5.15}$$

In order to compare the numerical results with the DNS data, two Re_τ were chosen for which DNS experiments were available [30]. They were $Re_\tau = 950$ and 2 000 and they corresponded to $Re = 37\,042$ and 86 734, respectively. The kinematic viscosity ν and the frictional velocity u_τ were accessible from the experiments [30]. In order to validate the FPSV-WF boundary condition with a higher Re, a third simulation for $Re_\tau = 16\,000$ was conducted. Since no DNS data were available for this case, the specific space-average velocity $u_{b,0}$ was chosen equal to the previous two cases, and Re was adjusted by setting ν accordingly. All the fluid properties of the three different Re_τ are given in Table 5.2.

All the simulations were ran with the cumulant LBM solver `LBMCumulantFoam` (Appendix B) as Implicit Large Eddy Simulation (ILES) without any explicit turbulence models. For the coarser grid $N10$, the time-step was $\Delta t_{N10} = 0.008\ s$ in order to have the maximum velocity in the bulk smaller than 0.1 in lattice units. For the other grids, the time-step was set by diffusive scaling of Δt_{N10} with respect to the cell size Δx ($\Delta t \propto \Delta x^2$). The values of the grid spacing Δx and time-step Δt for the three grids are shown in Table 5.1. The simulations were allowed to run over several number of channel passages before the analysis started. Data were obtained for the average of 60 channel passages, for a sample of nodes covering a line located at the middle of

Re_τ	Re	ν $[m^2/s]$ $\times 10^{-5}$	u_τ $[m/s]$	$\vec{u}_{b,0}$ $[m/s]$
950	37042	4.85908649173955	0.045390026	(0.9, 0, 0)
2 000	86734	2.06185567010309	0.041302030	(0.894, 0, 0)
16 000	933877	0.192771084337349	0.0308433734939759	(0.9, 0, 0)

Table 5.2: Summary of the computational set-up for the turbulent channel flow simulations.

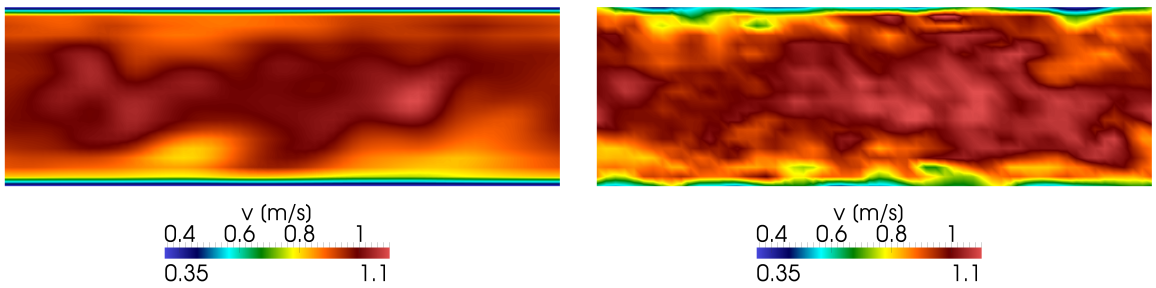
the channel from the bottom plane to the top plane. For averaging the channel was mirrored at the middle plane.

5.3.2 Relaxation parameters analysis

Two simulations were conducted in order to compare the new set of the relaxation parameters "Cum B" introduced in Chapter 4 with the standard set "Cum A" for turbulent flows. The very coarse grid $N10$ together with the Musker law method for the FPSV-WF and the lower frictional Reynolds number $Re_\tau = 950$ was used. Due to the coarse resolution of the mesh, the "Cum B" set needed a limiter coefficient $c_{lim} = 0.45$ in order to obtain a stable solution ("Cum B-lim").

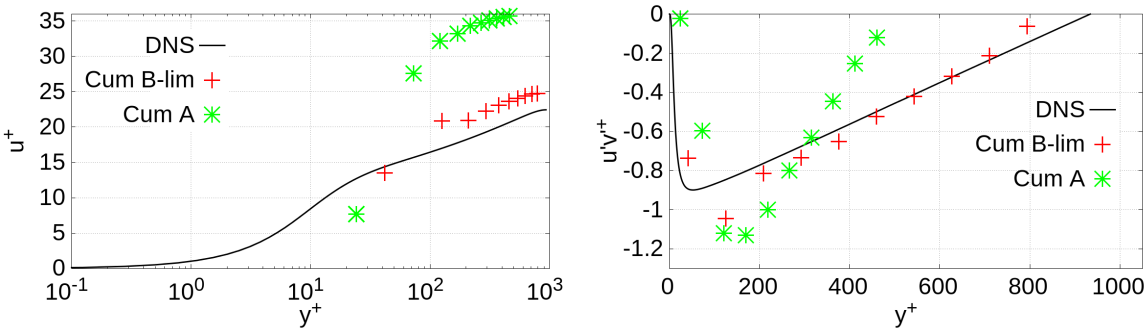
The instantaneous velocity colour plot on a slice at the centre of the channel showed large differences in the eddies size and turbulence intensity for the two sets of the relaxation parameters (Figure 5.5). With the standard set "Cum A", the boundary layer did not become turbulent (Figure 5.5a). The "Cum B-lim" set produced an highly turbulent flows even with a very coarse grid (Figure 5.5b). The flow field in the bulk was greatly different, being smoother for "Cum A" and more turbulent for "Cum B-lim".

The above results were confirmed by plotting the normalized velocity profiles and the normalized Reynold shear stress profiles in the direction normal to the planes (Figure 5.6). The relaxation parameter "Cum A" showed a lower y^+ at the first grid point and a higher velocity profile in comparison to the DNS data ($*$ points in Figure 5.6a). The "Cum B-lim" set showed a reasonable agreement with the DNS results ($+$ points in Figure 5.6a). The same observations held for the Reynold shear stress



(a) "Cum A" (b) "Cum B-lim"

Figure 5.5: Channel flow "Cum A" *vs* "Cum B-lim" - velocity colour plot. Instantaneous velocity colour plot on a slice at the centre of the channel for the coarser resolution grid ($N10$) at $Re_\tau = 950$. Simulations were conducted with the Musker law FPSV-WF by using the sets of the relaxation parameters "Cum A" (a) and "Cum B-lim" (b).



(a) Normalized velocity profile

(b) Reynolds shear stress

Figure 5.6: Channel flow "Cum A" *vs* "Cum B-lim" - results. Normalized velocity profile (a) and Reynolds shear stress (b) for the coarser resolution grid ($N10$) at $Re_\tau = 950$. Simulations were conducted with the Musker law FPSV-WF by using the sets of the relaxation parameters "Cum A" and "Cum B-lim".

profiles. While the "Cum B-lim" set showed a good agreement with the DNS data (+ points in Figure 5.6b), the "Cum A" set gave considerably lower stresses close to wall ($*$ points in Figure 5.6b), thus confirming the lower turbulence intensity of this set of relaxation parameters.

Since the set of the relaxation parameters "Cum B" showed proper results, all the following channel flow simulations were conducted with the "Cum B" set. Table 5.3 reports the "Cum B" settings for each simulation configuration. Interestingly, with only 20 grid nodes in the half channel eight ($N20$), the simulation at Re_τ 950 was performed without limiter coefficient.

5.3.3 Normalized velocity profiles

The cases for the two grid resolutions $N10$ and $N20$ and the three Reynolds numbers (950, 2 000, and 16 000) were simulated with the Musker law FPSV-WF and the D2V FPSV-WF. The grid $N40$ was also simulated for the three Reynolds numbers but only with the D2V FPSV-WF.

	$N10$	$N20$	$N40$
950	Cum B-lim (c_{lim}=0.45)	Cum B	Cum B
2 000	Cum B-lim (c_{lim}=0.3)	Cum B-lim (c_{lim}=2.25)	Cum B
16 000	Cum B-lim (c_{lim}=0.25)	Cum B-lim (c_{lim}=1)	Cum B-lim (c_{lim}=1)

Table 5.3: Summary of the sets of the relaxation parameters for the turbulent channel flow simulations.

The instantaneous velocity colour plot for the grid $N20$ at the three various Reynolds numbers is shown in Figure 5.7. The eddy sizes were largest for the lowest $Re_\tau = 950$ decreasing with higher Re (Figures 5.7a, 5.7c, and 5.7e). For the higher $Re_\tau = 16\,000$ the surface close to the top wall showed only few spots of low velocities (Figure 5.7f), resulting in a thin boundary layer thickness δ. With lower Reynolds numbers the zones of small velocities became larger, and therefore δ was bigger (Figures 5.7d and 5.7b).

In order to have quantitative results for the velocity field, the normalized velocity profiles for all the simulations were measured (Figure 5.8). The numerical results were compared to the DNS data [30] for the first two frictional Reynolds numbers $Re_\tau = 950$ and $2\,000$, while for the higher $Re_\tau = 16\,000$ the Musker law is used as reference.

The Musker law FPSV-WF showed reasonable results for all the Reynolds numbers and grid resolutions. Interestingly, the first point of the profile agreed well with the DNS data independently on the grid size (Figures 5.8a, 5.8c, and 5.8e). The rest of the profiles for the grid $N20$ was closer to the reference than for the coarser grid $N10$, thus showing a certain grid convergence of the results. Moreover, the grid $N10$ showed an evident kink at the second grid point while for the grid $N20$ this kink was smaller.

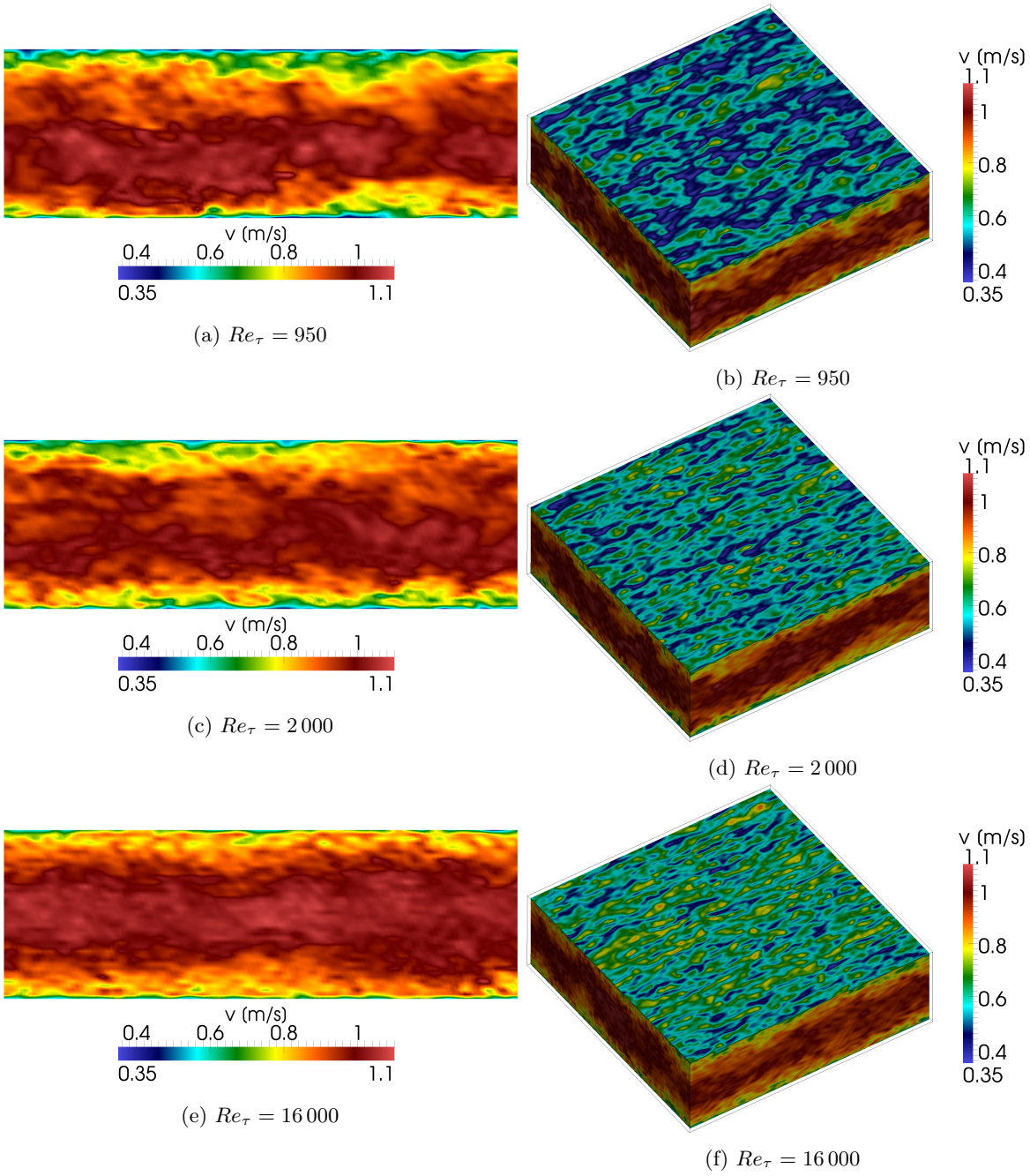
The D2V FPSV-WF showed larger discrepancies in the velocity profiles (Figures 5.8b, 5.8d, and 5.8f). With the very coarse mesh $N10$ the wall function overestimated the wall shear stress τ_w and consequently u_τ for all the Reynolds numbers, and the velocities were lower than in the DNS data. By increasing the resolution with the grid $N20$, the velocity profiles for the $Re_\tau = 950$ and $2\,000$ had a reasonable agreement with the DNS data, while for the higher $Re_\tau = 16\,000$ the velocities were lower. With the grid $N40$, the velocity profiles for the $Re_\tau = 2\,000$ and $16\,000$ had a better agreement with the DNS data than the grid $N20$, while for the lower $Re_\tau = 950$ did not.

5.3.4 Normalized Reynold shear stress profiles

The normalized Reynold shear stress $u'v'^+$ was computed as the product of the fluctuating parts of the velocity field in the stream-wise and normal directions:

$$\begin{aligned} u'v'^+ &= uv^+ - \bar{u}^+\bar{v}^+, \\ uv^+ &= \frac{1}{n_t}\sum_{t=t_s}^{t_e}\frac{uv}{u_\tau^2}, \quad \bar{u}^+ = \frac{1}{n_t}\sum_{t=t_s}^{t_e}\frac{u}{u_\tau}, \quad \bar{v}^+ = \frac{1}{n_t}\sum_{t=t_s}^{t_e}\frac{v}{u_\tau}, \end{aligned} \tag{5.16}$$

where t_s and t_e were the initial and final averaging time-steps, respectively, and n_t was the total number of time-steps used for the averaging process. Figure 5.9 shows the Reynold shear stress profiles for all the simulations.



(a) $Re_\tau = 950$

(b) $Re_\tau = 950$

(c) $Re_\tau = 2\,000$

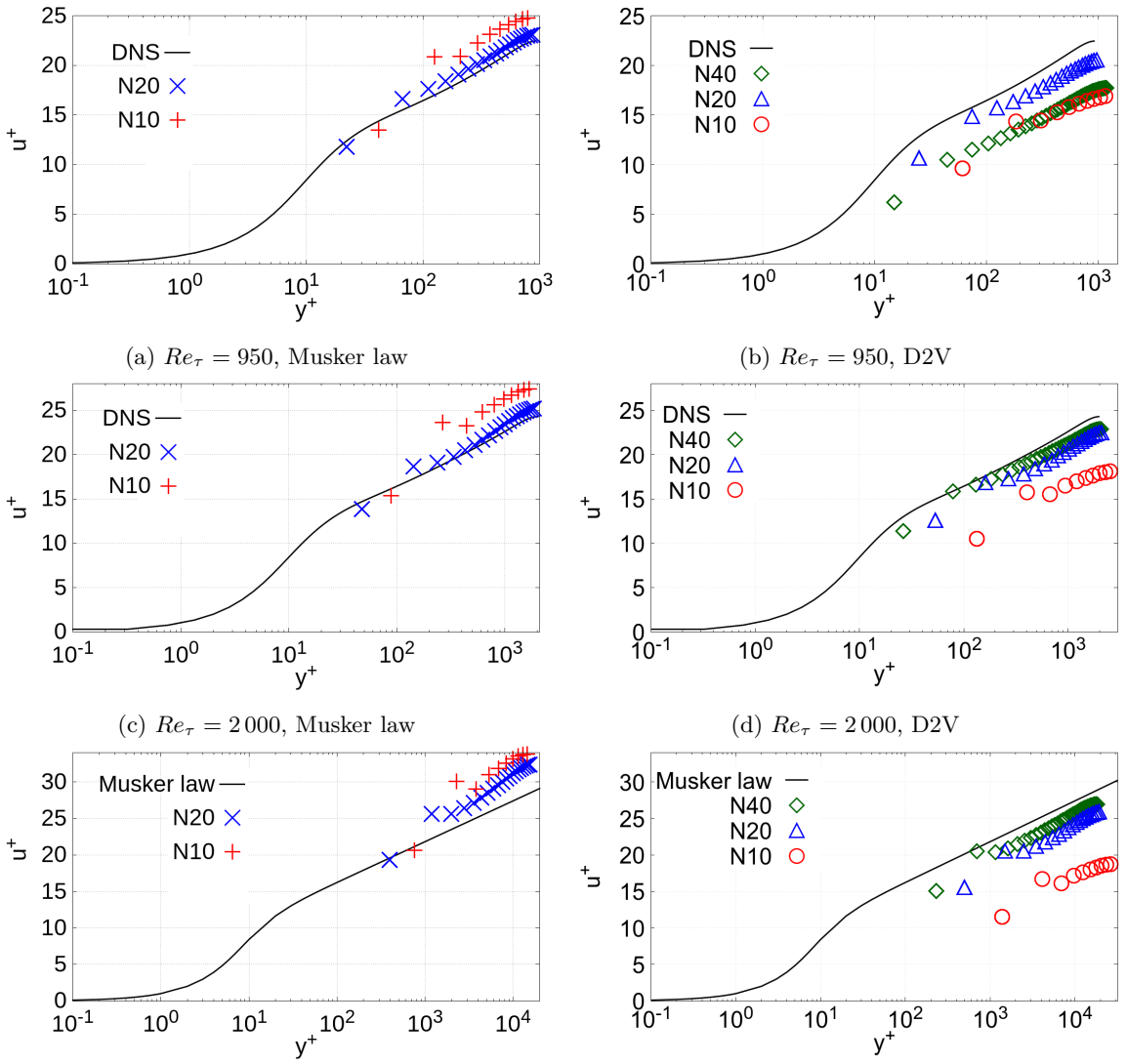
(d) $Re_\tau = 2\,000$

(e) $Re_\tau = 16\,000$

(f) $Re_\tau = 16\,000$

Figure 5.7: Channel flow - velocity colour plot. Instantaneous velocity colour plot in the channel domain for the grid $N20$ at different Re_τ by using the Musker law FPSV-WF. Slice at the centre of the channel (a, c, e) and isometric view (b, d, f).

The Musker law FPSV-WF showed reasonable agreement for $Re_\tau = 950$ and $2\,000$ for both grids. For the grid $N10$, the kink in the velocity profile at the second grid point was also visible in the Reynolds shear stress, having a larger value than in the



(a) $Re_\tau = 950$, Musker law

(b) $Re_\tau = 950$, D2V

(c) $Re_\tau = 2\,000$, Musker law

(d) $Re_\tau = 2\,000$, D2V

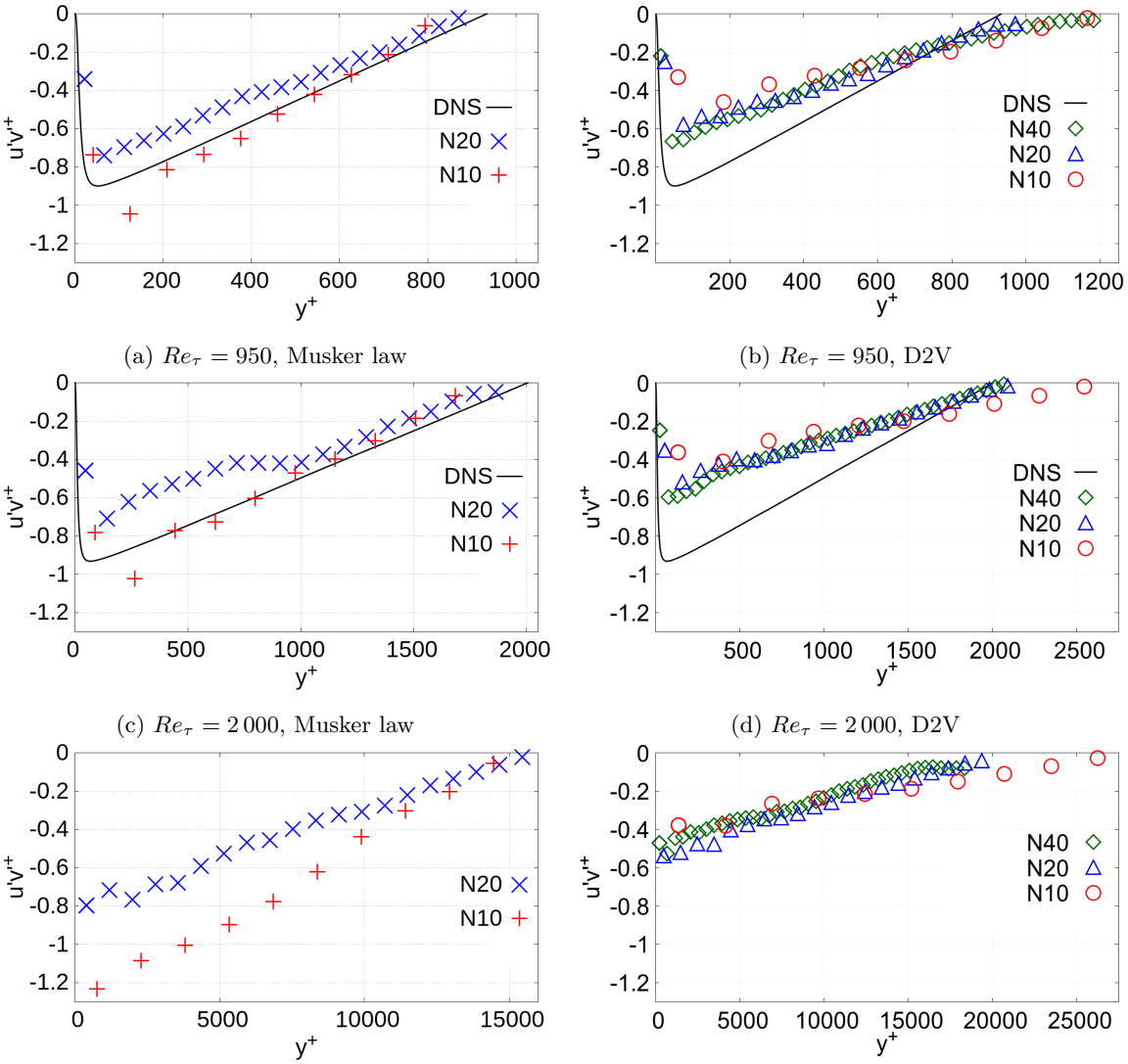
(e) $Re_\tau = 16\,000$, Musker law

(f) $Re_\tau = 16\,000$, D2V

Figure 5.8: Channel flow - normalized velocity profile. Normalized velocity profile for different grid resolutions, Re_τ, and FPSV-WF methods.

DNS data. This affected all the rest of the profile, producing a larger slope of the curve than in the DNS data. The grid $N20$ showed a wide bump in the profile close to the wall for $Re_\tau = 2000$.

The D2V FPSV-WF showed large discrepancies in the profiles for $Re_\tau = 950$ and $2\,000$. The coarser grid $N10$ over predicted τ_w and u_τ, and the normalized Reynolds shear stress had profiles with smaller values and lower slopes than in the DNS data. With the less coarse grid $N20$ the situation improved, τ_w and u_τ decreased and the normalized Reynolds shear stress were closer to the reference, especially in the bulk



(a) $Re_\tau = 950$, Musker law

(b) $Re_\tau = 950$, D2V

(c) $Re_\tau = 2\,000$, Musker law

(d) $Re_\tau = 2\,000$, D2V

(e) $Re_\tau = 16\,000$, Musker law

(f) $Re_\tau = 16\,000$, D2V

Figure 5.9: Channel flow - Reynold shear stress. Reynold shear stress for different grid resolutions, Re_τ, and FPSV-WF methods.

of the domain. For the grid $N40$, the normalized Reynolds shear stress profiles for $Re_\tau = 2\,000$ and $16\,000$ were similar to those ones of grid $N20$. For $Re_\tau = 950$, the profile was similar to that one of grid $N20$ close to the wall and to that one of grid $N10$ at the centre of the channel.

For the higher $Re_\tau = 16\,000$, no reference data were available for comparison. Nevertheless, their trends were in accordance with the profiles at lower Reynolds numbers for both the FPSV-WFs.

5.4 Discussion and summary

The frictional partial slip velocity wall function (FPSV-WF) has been tested with turbulent channel flow simulations at different Reynolds numbers with very coarse grids. It is important to remark that coarse grids were used with the intention to stress the wall function.

The Musker law FPSV-WF over predicted the normalized velocity and Reynolds shear stress profiles in the channel for all the Reynolds numbers tested (Figures 5.8a, 5.8c, 5.8e, 5.9a, 5.9c, and 5.9e). The reason could be due to the large kink at the second grid point, leading to the over estimation of the rest of the profile. While the velocity of the first grid point was enforced by the wall function, the second grid point was free. Therefore, the velocity at the second grid point depended on the grid resolution while at the first one it did not. The coarser grid $N10$ had a larger kink at the second grid node than the grid $N20$. The first grid point was also the boundary node where the wall function computed the wall shear stress applying the slip velocity. Therefore, the wall function did not depend on the grid resolution and both grids $N10$ and $N20$ gave results in a reasonable agreement with the DNS data.

The D2V FPSV-WF showed discrepancies in all the profiles (Figures 5.8b, 5.8d, 5.8f, 5.9b, 5.9d, and 5.9f). The reason could be due to the method chosen for solving the second derivative of the velocity of Eq. (5.8) for obtaining the wall shear stress τ_w. The method computed a finite difference between the fluid node and the position of the wall. While this approach gave correct results of τ_w for the Poiseuille flow at low Reynolds number, it was strongly dependent on the grid resolution for turbulent flows at high Reynolds number. Especially for the coarser grid $N10$, τ_w was over estimated, leading to a lower slip velocity at the wall and to a lower position of the first grid point in the profiles in comparison to the reference data. By increasing the resolution with the grid $N20$, the error of the computed τ_w was lower, and all the profiles were in reasonable agreement with the DNS data, especially for the two lower $Re_\tau = 950$ and 2 000. For the higher $Re_\tau = 16\,000$ the error was still high. The finer grid $N40$ showed mixed results. For the lower $Re_\tau = 950$ the results were worse than for the grid $N20$, while for the other two Reynolds numbers were better. A first reason can be that the data were obtained for a smaller number of channel passages than for the grid $N20$. Another reason can be that, for $Re_\tau = 950$, the y^+ of the first grid point was very close to the value of the intersection between the linear and the logarithmic profile of the "law of the wall" ($y^+ \approx 11$). This region is the zone where the transition from laminar to turbulent flow happens, which is a phenomenon very difficult to model.

Nevertheless, the method proposed in Eq. (5.8) had the advantage to use information from cumulants, which was available directly at the boundary nodes. Further work could be aimed at addressing the implementation of Gauss-Legendre integration [115] and approximation methods [109].

The new set of the relaxation parameters "Cum B" implemented for turbulent flows at high Re predicted the fluctuating parts of the velocity fields, while the standard set of the relaxation parameters "Cum A" did not (Figure 5.6b). Interestingly, "Cum B" gave a good agreement with the DNS data for the Reynolds shear stress by simulating without any explicit LES turbulence model. Hence, the choice of the set of the relaxation parameters was crucial in order to develop a proper turbulent flow for the FPSV-WF.

6 | Conclusion

That sun which erst had warmed my heart with love,
by proving and refuting, had revealed
to me the pleasing face of lovely truth;
Dante Alighieri, *The Divine Comedy, vol. 3 (Paradiso)*

Computational Fluid Dynamics (CFD) for complex engineering problems has been investigated by considering two aspects. They were the discretization of complex geometries and the analysis of turbulent flows, especially in the near-wall region. Since engineers deal with the design of complex objects such as cars, airplanes, buildings, porous media, etc., the numerical methods utilized for studying and analysing them have to be assisted by discretization techniques that are able to generate grids for modelling properly the real objects. Since in many engineering applications the fluid flow is mostly turbulent, the study of flows at high Reynolds numbers becomes necessary. Turbulent flows are difficult to investigate as many phenomena such as unsteadiness, instabilities, chaotic effects, etc., perturb the flow field.

This work addressed these two aspects by considering the cumulant lattice Boltzmann method (LBM) for carrying out CFD simulations [43]. Although being relatively new on the CFD panorama, the LBM became a valid alternative to standard CFD approaches such as the Finite Volume Method (FVM). The cumulant LBM showed to overcome some issues of the standard LBM, such as the violation of the Galilean invariance, the spurious coupling of the degrees of freedoms, and the hyper-viscosity. It has been successfully implemented for High Performance Computing (HPC) on CPU and GPGPU applications [65], showing accuracy of results and high performance [43, 66, 67, 90].

The discretization of the computational domain, although being considered trivial [44, 45], presents some challenges if applied to complex geometries using nested Cartesian grids. For the LBM method to be as attractive as other CFD methods, some special features have to be provided. These features are, for example, the second order boundary definition, and the grid refinement. Since the standard LBM requires Cartesian grids for the streaming procedure, boundaries have to set by cut-cell approach [63]. If solving fluid flows at high Re, the grid refinement is necessary to use fine grids

where required and coarse grids where possible [70]. In this work, a new grid generator that creates advanced grids for the cumulant LBM was introduced (Chapter 3). For defining complex boundaries, a technique that computes the sub-grid-distances for the cut-cell approach was implemented based on bounding boxes. Regarding the grid refinement, the "frame" data structure enables an efficient grid refinement interface definition. The grid refinement is not shape-constrained and it has level wise load balancing. The format of the data for describing the grid was optimized in order to maximize the number of grid points fitting in a given memory of a GPGPU, either for serial or parallel applications. During its development, `LBMHexMesh` was tested for generating grids with about 500 M grid points. Further developments, e.g. a new level wise load balancing, are required for the generation of very large meshes with more than 1 000 M grid points for applications on large multi-GPGPUs systems.

The turbulent flow in the near-wall region has been investigated by considering two aspects for the cumulant LBM. The first aspect was the analysis of the relaxation parameters for the third order cumulants, while the second one was the implementation of a wall model for treating the turbulence in the near-wall region.

The relaxation parameters for multiple relaxation time LBMs have been already demonstrated to be influential for the accuracy of the results, especially for bounded flows [46, 47]. However, the definition of these parameters represents a sort of compromise because the number of constrains is larger than the number of parameters. The new set of parameters used in this work was chosen such that the Linearized Leading Error decreases for small shear viscosities independent of the bulk viscosity (Chapter 4). Various validation cases were tested with the new parameters, such as the Taylor Green vortex, the shear wave, the Poiseuille flow, and the turbulent channel flow. The new set of relaxation parameters provided a proper turbulence intensity for turbulent channel flow even at very coarse resolutions.

Despite of being a relevant aspect of turbulent flows, the near-wall region turbulence treatment for the LBM has only recently been addressed [48]. Near-wall treatment is important as it allows to estimate the correct quantities at the wall, e.g. the stresses, even with coarse grids, thus reducing the number of grid points necessary in the boundary layer and increasing the efficiency of the numerical simulations. I proposed to solve the problem by developing a new wall function that needed only local information at the boundary nodes and that used quantities computed from cumulants (Chapter 5). By imposing the exact wall shear stress from the quasi-analytical solution at the boundary node, the Musker law FPSV-WF gave better results than the D2V FPSV-WF. The D2V FPSV-WF used Eq. (5.6) for solving the second derivative of the velocity. The

solution depended strongly on the grid resolution. Further work could be aimed at addressing the computation of the second derivative of the velocity by a more accurate method. For its application on curved geometries such as spheres or cars, the second derivate in wall normal direction of the velocity tangential to the wall can not be calculated with the cumulants of the D3Q27 lattice. The reason is that the cumulants of third order C_{300}, C_{030}, and C_{003} are missing. A new lattice that supports all ten third order cumulants definition can be used, e.g. the Body Centered Cubic (BCC) lattice [118].

In conclusion, since the treated topic is complex and broad, this work can be considered as an initial step rather than an exhaustive attempt for finding the solutions of the addressed problems. Indeed, as highlighted by the quote above, science is based on the concept of "proving and refuting", experimenting and confirming [119, 120]. Only with this approach the scientific knowledge can grow.

A | Cumulant LBM kernel

The D3Q27 cumulant LBM collision operator is implemented by performing five steps [43].

The first step is the "forward central moments transformation", which computes the central moments from the distributions. The central moments computation is split among the three different directions ijk:

$$K_{ij|\gamma} = \sum_k f_{ijk}(k - w/c)^\gamma, \tag{A.1}$$

$$K_{i|\beta\gamma} = \sum_j K_{ij|\gamma}(j - v/c)^\beta, \tag{A.2}$$

$$K_{\alpha\beta\gamma} = \sum_i K_{i|\beta\gamma}(i - u/c)^\alpha, \tag{A.3}$$

where u, v, w are computed with Eq. (2.6), $c = \Delta x/\Delta t$, and $\alpha, \beta, \gamma = 0 \cdots 2$.

The second step is the "forward cumulants transformation", which computes the cumulants from the central moments. Since the equilibrium of most cumulants is zero the normalization is omitted in the following [43]. The cumulants until the third order are indentical to the central moments:

$$C_{110} = K_{110}, \quad C_{200} = K_{200}, \quad C_{120} = K_{120}, \quad C_{111} = K_{111}. \tag{A.4}$$

The cumulants differ from central moments from fourth order:

$$\begin{aligned}
C_{211} &= K_{211} - (K_{200}K_{011} + 2K110K_{101})/\rho, \\
C_{220} &= K_{220} - (K_{200}K_{020} + 2K110^2)/\rho, \\
C_{122} &= K_{122} - (K_{002}K_{120} + K_{020}K_{102} + 4K_{011}K_{111} + 2(K_{101}K_{021} + K_{110}K_{012}))/\rho, \\
C_{222} &= K_{222} \\
&\quad - (4K_{111}^2 + K_{200}K_{022} + K_{020}K_{202} + K_{002}K_{220} + 4(K_{011}K_{211} + K_{101}K_{121} + K_{110}K_{112}) \\
&\quad + 2(K_{120}K_{102} + K_{210}K_{012} + K_{201}K_{021}))/\rho \\
&\quad + (16K_{110}K_{101}K_{011} + 4(K_{101}^2K_{020} + K_{011}^2K_{200} + K_{110}^2K_{002}) + 2K_{200}K_{020}K_{002})/\rho^2.
\end{aligned} \tag{A.5}$$

The omitted cumulants can be obtained by permuting the indices.

The third step is the "collision". The collision reads explicitly:

$$\begin{aligned}
C^*_{110} &= (1-\omega_1)C_{110}, \quad C^*_{101} = (1-\omega_1)C_{101}, \quad C^*_{011} = (1-\omega_1)C_{011}, \\
C^*_{200} - C^*_{020} &= (1-\omega_1)(C_{200} - C_{020}) - 3\rho\left(1 - \frac{\omega_1}{2}\right)(u^2 D_x u - v^2 D_y v), \\
C^*_{200} - C^*_{002} &= (1-\omega_1)(C_{200} - C_{002}) - 3\rho\left(1 - \frac{\omega_1}{2}\right)(u^2 D_x u - w^2 D_z w), \\
C^*_{200} + C^*_{020} + C^*_{002} &= K_{000}\omega_2 + (1-\omega_2)(C_{200} + C_{020} + C_{002}) \\
&\quad - 3\rho\left(1 - \frac{\omega_2}{2}\right)(u^2 D_x u + v^2 D_y v + w^2 D_z w),
\end{aligned} \tag{A.6}$$

where $D_x u$, $D_y v$, and $D_z w$ are computed as:

$$\begin{aligned}
D_x u &= -\frac{\omega_1}{2\rho}(2C_{200} - C_{020} - C_{002}) - \frac{\omega_2}{2\rho}(2C_{200} + C_{020} + C_{002} - K_{000}), \\
D_y v &= D_x u + \frac{3\omega_1}{2\rho}(C_{200} - C_{020}), \\
D_z w &= D_x u + \frac{3\omega_1}{2\rho}(C_{200} - C_{002}).
\end{aligned} \tag{A.7}$$

The equilibria for all remaining cumulants are zero:

$$\begin{aligned}
C^*_{120} + C^*_{102} &= (1-\omega_3)(C_{120} + C_{102}), \\
C^*_{210} + C^*_{012} &= (1-\omega_3)(C_{210} + C_{012}), \\
C^*_{201} + C^*_{021} &= (1-\omega_3)(C_{201} + C_{021}), \\
C^*_{120} - C^*_{102} &= (1-\omega_4)(C_{120} - C_{102}), \\
C^*_{210} - C^*_{012} &= (1-\omega_4)(C_{210} - C_{012}), \\
C^*_{201} - C^*_{021} &= (1-\omega_4)(C_{201} - C_{021}), \\
C^*_{111} &= (1-\omega_5)C_{111},
\end{aligned} \tag{A.8}$$

$$\begin{aligned}
C^*_{220} - 2C^*_{202} + C^*_{022} &= (1-\omega_6)(C_{220} - 2C_{202} + C_{022}), \\
C^*_{220} + C^*_{202} - 2C^*_{022} &= (1-\omega_6)(C_{220} + C_{202} - 2C_{022}), \\
C^*_{220} + C^*_{202} + C^*_{022} &= (1-\omega_7)(C_{220} + C_{202} + C_{022}), \\
C^*_{211} &= (1-\omega_8)C_{211}, \\
C^*_{121} &= (1-\omega_8)C_{121}, \\
C^*_{112} &= (1-\omega_8)C_{112},
\end{aligned} \tag{A.9}$$

$$\begin{aligned} C^*_{221} &= (1-\omega_9)C_{221}, \\ C^*_{212} &= (1-\omega_9)C_{212}, \\ C^*_{122} &= (1-\omega_9)C_{122}, \end{aligned} \tag{A.10}$$

$$C^*_{222} = (1-\omega_{10})C_{222}. \tag{A.11}$$

The fourth step is the "backward cumulants transformation", which recomputes the central moments from the cumulants after collision:

$$\begin{aligned} K^*_{211} &= C^*_{211} - (K^*_{200}K^*_{011} + 2K^*_{110}K^*_{101})/\rho, \\ K^*_{220} &= C^*_{220} - (K^*_{200}K^*_{020} + 2{K^*_{110}}^2)/\rho, \\ K^*_{122} &= C^*_{122} - (K^*_{002}K^*_{120} + K^*_{020}K^*_{102} + 4K^*_{011}K^*_{111} + 2(K^*_{101}K^*_{021} + K^*_{110}K^*_{012}))/\rho, \\ K^*_{222} &= C^*_{222} \\ &\quad - (4{K^*_{111}}^2 + K^*_{200}K^*_{022} + K^*_{020}K^*_{202} + K^*_{002}K^*_{220} + 4(K^*_{011}K^*_{211} + K^*_{101}K^*_{121} + K^*_{110}K^*_{112}) \\ &\quad + 2(K^*_{120}K^*_{102} + K^*_{210}K^*_{012} + K^*_{201}K^*_{021}))/\rho \\ &\quad + (16K^*_{110}K^*_{101}K^*_{011} + 4({K^*_{101}}^2K^*_{020} + {K^*_{011}}^2K^*_{200} + {K^*_{110}}^2K^*_{002}) + 2K^*_{200}K^*_{020}K^*_{002})/\rho^2. \end{aligned} \tag{A.12}$$

The remaining central moments can be obtained by permuting the indices.

The fifth step is the "backward central moments transformation", which recomputes the distributions from the central moments after collision:

$$K^*_{0|\beta\gamma} = K^*_{0\beta\gamma}(1-(u/c)^2) - 2(u/c)K^*_{1\beta\gamma} - K^*_{2\beta\gamma}, \tag{A.13}$$

$$K^*_{\bar{1}|\beta\gamma} = (K^*_{0\beta\gamma}((u/c)^2 - u/c) + K^*_{1\beta\gamma}(2u/c-1) + K^*_{2\beta\gamma})/2, \tag{A.14}$$

$$K^*_{1|\beta\gamma} = (K^*_{0\beta\gamma}((u/c)^2 + u/c) + K^*_{1\beta\gamma}(2u/c+1) + K^*_{2\beta\gamma})/2, \tag{A.15}$$

$$K^*_{i0|\gamma} = K^*_{i|0\gamma}(1-(v/c)^2) - 2(v/c)K^*_{i|1\gamma} - K^*_{i|2\gamma}, \tag{A.16}$$

$$K^*_{i\bar{1}|\gamma} = (K^*_{i|0\gamma}((v/c)^2 - v/c) + K^*_{i|1\gamma}(2v/c-1) + K^*_{i|2\gamma})/2, \tag{A.17}$$

$$K^*_{i1|\gamma} = (K^*_{i|0\gamma}((v/c)^2 + v/c) + K^*_{i|1\gamma}(2v/c+1) + K^*_{i|2\gamma})/2, \tag{A.18}$$

$$f^*_{ij0} = K^*_{ij|0}(1-(w/c)^2) - 2(w/c)K^*_{ij|1} - K^*_{ij|2}, \tag{A.19}$$

$$f^*_{ij\bar{1}} = (K^*_{ij|0}((w/c)^2 - w/c) + K^*_{ij|1}(2w/c-1) + K^*_{ij|2})/2, \tag{A.20}$$

$$f^*_{ij1} = (K^*_{ij|0}((w/c)^2 + w/c) + K^*_{ij|1}(2w/c+1) + K^*_{ij|2})/2. \tag{A.21}$$

The collision step is followed by the usual streaming step.

B | LBMCumulantFoam

`LBMCumulantFoam` is a cumulant LBM implementation within the OpenFOAM environment [72], following the work started with `LBMHexMesh` (Chapter 3). The settings for running simulations with `LBMCumulantFoam` are divided into three types, boundary conditions type, properties type, and control type.

The boundary condition settings have already been shown in Listing 3.8.

The properties settings are described in the file `transportProperties` (Listing B.1). The default kernel used in `LBMCumulantFoam` is the cumulant LBM. Nevertheless, it is possible to specify also the BGK kernel by declaring the optional entry `kernel` as `BGK`. The default initialization of the fluid is with a zero velocity for all the grid nodes. In order to allow other initializations of the fluid flow, the optional entry `init` must be declared. The other initialization possibilities are the shear wave `ShearWave` and the Taylor Green vortex `TaylorGreen`. The kinematic viscosity and the fluid density are set in the entries `nu` and `rho`, respectively, and the values must be in International System of Units (SI). For the computation of some quantities

```
//kernel BGK; // default=cumulant

//init ShearWave; // ShearWave // TaylorGreen // default=velocity (0 0 0)

nu 1.51E-005; // kinematic viscosity [m2/s]
rho 1.0; // density [kg/m3]
useForCalc mean; // mean // eff

L 2.0; // reference length [m]
UBulk 0.9; // reference velocity
ReBulk 0.0; // Reynolds number
Dir ( 1 0 0 ); // main flow direction

LESmodel    None; // None // Smagorinsky
// Smagorinsky model constants
kappa       0.410; // von Karman constant
E           9.8;
Cs          0.08; // Smagorinsky coeff.
filterWidth cellSize; // filter width
Cvd         26.0; // van Driest coeff.

relParThirdMom  Cum_B-lim; // rel. par. set
coeff           0.45; // limiter coeff.
```

Listing B.1: `transportProperties` file example.

such as stresses and Reynolds number, one can choose to use averaged values or local values. This is possible by setting the entry `useForCalc` as `mean` or `eff` for using averaged or local values, respectively. Other useful parameters are the reference length `L`, the reference velocity `UBulk`, the Reynolds number `ReBulk`, and the main flow direction `Dir`. All are in SI units. If at least one of velocity or pressure boundary conditions is used, these entries are ignored. If neither velocity nor pressure boundary conditions are specified, these entries are used for setting a forcing term in the bulk. For example:

- fixed forcing term, with `UBulk` = `ReBulk` > 0 being the magnitude of the forcing term and `Dir` being the direction,
- velocity-adaptive forcing term like in Eq. (5.14), with `ReBulk` $= 0$ and `Dir` being the direction. The force adapts in order to have the mean velocity in the bulk as specified in `UBulk` > 0,
- *Re*-adaptive forcing term, with `UBulk` $= 0$ and `Dir` being the direction. The force adapts in order to have the specified Reynolds number `ReBulk` > 0.

LES computations can be approached with either implicit or explicit models. The entry to use is `LESmodel` and it can be `None` or `Smagorinsky`, respectively. For the latter case some constants have to be provided (see Listing B.1). The last entry `relParThirdMom` is about the set of relaxation parameters for the third order cumulants. The sets can be `Cum_A`, `Cum_B`, and `Cum_B-lim` (Chapter 4). If the bounded version `Cum_B-lim` is used, the limiter coefficient c_{lim} has to be provided by setting the entry `coeff`.

The control settings are used for describing the evolution of the simulation and the functions for extracting quantities during the computation (Listing B.2). All the time values specified in this file are in seconds, since the entry `writeControl` is set as `runTime`. The simulation starts from time 0 if no restart file is found, and it stops at the time described in the `endTime` entry. The time-step Δt is set in the entry `deltaT` and the results can be written in the specified intervals. VTK files are written every `writeInterval` intervals, restart files every `writeRestart` intervals, and the values requested by the functions every `writeProbe` intervals. Mean values can be printed every `writeMean` intervals. The files can be written either in binary or ASCII format by setting the entry `writeFormat` as `binary` or `ascii`, respectively. For parallel executions, `LBMCumulantFoam` uses the Open Multi-Processing (OpenMP) application and the number of threads for the simulation can be set in

```
application     LBMCumulantFoam;

writeControl    runTime; // time in seconds
stopAt          endTime;
endTime          500000; // last time
deltaT                0.002; // time-step
writeInterval        10.000; // write VTK
writeRestart         10.0;   // write restart file
writeProbe            0.002; // write values from 'Functions'
writeMean             0.002; // show mean values

writeFormat binary; // binary // ascii
memInfo     on; // on // default=off
nProcs      8; // n. threads for parallel execution

// FOAM parameters // do not change
startFrom        latestTime;
startTime        0;
timeFormat       general;
timePrecision    16;
writePrecision   16;
writeCompression compressed;
purgeWrite       0;

// Functions
extractType            Line;
extract_Line_origin    (0 0 0);
extract_Line_direction 4;

extractType        Patch;
extract_Patch_name geom;
```

Listing B.2: `controlDict` file example.

the `nProcs` entry. Finally, in order to extract values from the computation at specific locations in the domain, two different functions are available by declaring the entry `extractType`. The first function allows to collect information over a sample of points along a specified line (`extractType` is `Line`). The line starts from a specified point of the domain and proceeds along one of the 26 directions of the lattice D3Q27. All the points sampled are physical nodes of the computational grid. The origin and the direction of the line are specified with the entries `extract_Line_origin` and `extract_Line_direction`, respectively. The direction is an integer number that identifies the link of the lattice D3Q27 source node, e.g. 4 is the direction in +Z, and 26 is the direction in -Z, -Y, -X. The second function extracts the integral values for the pressure and velocity fields over a boundary (`extractType` is `Patch`). The name of the boundary patch has to be defined with the entry `extract_Patch_name` and it must be an existing boundary of the mesh.

C | Asymptotic analysis until third order

Eq. (5.6) was derived after performing a combination of Taylor expansion [105] and asymptotic analysis [106] of Eq. (2.4) until the third order in diffusive scaling [43]. The lhs of Eq. (2.4) can be Taylor expanded in time, the rhs can be Taylor expanded in space:

$$\sum_{o=0}^{\infty} \frac{\Delta t^o}{o!} \partial_{t^o} f_{ijkxyzt} = \sum_{m,n,l=0}^{\infty} \frac{(i^m j^n k^l)(-c\Delta t)^{m+n+l}}{m!n!l!} \partial_{x^m y^n z^l} f^*_{ijkxyzt}. \tag{C.1}$$

By inserting the moments $m_{\alpha\beta\gamma} = \sum_{ijk} i^\alpha j^\beta k^\gamma f_{ijk}$, it is possible to write:

$$\sum_{o=0}^{\infty} \frac{\Delta t^o}{o!} \partial_{t^o} m_{\alpha\beta\gamma} = \sum_{m,n,l=0}^{\infty} \frac{(-c\Delta t)^{m+n+l}}{m!n!l!} \partial_{x^m y^n z^l} m^*_{(\alpha+m)(\beta+n)(\gamma+l)}. \tag{C.2}$$

The expansion in moments can be used also with the cumulant collision operator since cumulants can be transformed into moments. Time and space variables are substituted by dimensionless ones by adopting diffusive scaling: $\Delta t \propto \epsilon^2$ and $\Delta x \propto \epsilon$, with $c\Delta t = \Delta x$. The term ϵ is the scaling parameter. Eq. (C.2) becomes:

$$\sum_{o=0}^{\infty} \frac{\epsilon^{2o}}{o!} \partial_{t^o} m_{\alpha\beta\gamma} = \sum_{m,n,l=0}^{\infty} \frac{(-\epsilon)^{m+n+l}}{m!n!l!} \partial_{x^m y^n z^l} m^*_{(\alpha+m)(\beta+n)(\gamma+l)}. \tag{C.3}$$

The moments are expanded asymptotically in ϵ:

$$m_{\alpha\beta\gamma} = \sum_{q=0}^{\infty} \epsilon^q m^{(q)}_{\alpha\beta\gamma}, \quad m^*_{\alpha\beta\gamma} = \sum_{q=0}^{\infty} \epsilon^q m^{*(q)}_{\alpha\beta\gamma}, \tag{C.4}$$

and Eq. (C.3) becomes:

$$\sum_{o=0}^{\infty} \frac{\epsilon^{2o}}{o!} \partial_{t^o} \sum_{q=0}^{\infty} \epsilon^q m^{(q)}_{\alpha\beta\gamma} = \sum_{m,n,l=0}^{\infty} \frac{(-\epsilon)^{m+n+l}}{m!n!l!} \partial_{x^m y^n z^l} \sum_{q=0}^{\infty} \epsilon^q m^{*(q)}_{(\alpha+m)(\beta+n)(\gamma+l)}. \tag{C.5}$$

Since the derivation is done for the D3Q27 lattice, only 27 moments are independent. The higher order moments are considered by aliasing condition:

$$m_{300} = m_{100}, \quad m_{400} = m_{200}, \quad m_{500} = m_{100}, \quad m_{310} = m_{110} \tag{C.6}$$

and so on. The moments are collision invariant at zeroth order ϵ^0 and first order ϵ^1 (no proof) [43]:

$$m^{(0)}_{\alpha\beta\gamma} = m^{*(0)}_{\alpha\beta\gamma}, \quad m^{(1)}_{\alpha\beta\gamma} = m^{*(1)}_{\alpha\beta\gamma} = m^{eq(1)}_{\alpha\beta\gamma}. \tag{C.7}$$

The relationship between the third order cumulants and the second derivative of the velocity is obtained at third order ϵ^3. For obtaining the second derivative in Y of the velocity in X ($\partial_{yy}u$), it is possible to write ($\alpha = 1$, $\beta = 2$, and $\gamma = 0$):

$$\begin{aligned} m^{(3)}_{120} + \partial_t m^{(1)}_{120} = m^{*(3)}_{120} - \partial_x m^{*(2)}_{220} - \partial_y m^{*(2)}_{110} - \partial_z m^{*(2)}_{121} \\ + 1/2(\partial_{xx} m^{*(1)}_{120} + \partial_{yy} m^{*(1)}_{120} + \partial_{zz} m^{*(1)}_{122}) \\ + \partial_{xy} m^{*(1)}_{210} + \partial_{xz} m^{*(1)}_{221} + \partial_{yz} m^{*(1)}_{111}. \end{aligned} \tag{C.8}$$

Due to Eq. (C.7), the moments of first order ϵ^1 can be written as ($\theta = 1/3$ is the dimensionless speed of sound squared):

$$m^{*(1)}_{120} = \theta m^{(1)}_{100} = 1/3 \rho^{(0)} u^{(1)}, \tag{C.9}$$

$$m^{*(1)}_{122} = \theta^2 m^{(1)}_{100} = 1/9 \rho^{(0)} u^{(1)}, \tag{C.10}$$

$$m^{*(1)}_{210} = \theta m^{(1)}_{010} = 1/3 \rho^{(0)} v^{(1)}, \tag{C.11}$$

$$m^{*(1)}_{221} = \theta^2 m^{(1)}_{001} = 1/9 \rho^{(0)} w^{(1)}. \tag{C.12}$$

$$m^{*(1)}_{111} = 0, \tag{C.13}$$

The moment of second order $m^{*(2)}_{110}$ can be written as:

$$m^{*(2)}_{110} = (1 - \omega_1) m^{(2)}_{110} + \omega_1 m^{eq(2)}_{110}, \tag{C.14}$$

with $m^{(2)}_{110}$ [43]:

$$m^{(2)}_{110} = m^{*(2)}_{110} - (\partial_x m^{eq(1)}_{210} + \partial_y m^{eq(1)}_{120} + \partial_z m^{eq(1)}_{111}). \tag{C.15}$$

The following terms are introduced:

$$\begin{aligned} m_{110}^{eq(2)} &= \rho^{(0)} u^{(1)} v^{(1)}, \\ m_{210}^{eq(1)} &= \theta \rho^{(0)} v^{(1)} = 1/3 \rho^{(0)} v^{(1)}, \\ m_{120}^{eq(1)} &= \theta \rho^{(0)} u^{(1)} = 1/3 \rho^{(0)} u^{(1)}, \\ m_{111}^{eq(1)} &= 0. \end{aligned} \tag{C.16}$$

By inserting Eqs. (C.15) and (C.16) into Eq. (C.14), it is possible to write:

$$m_{110}^{*(2)} = \frac{(1-\omega_1)}{\omega_1}(-\partial_x 1/3 \rho^{(0)} v^{(1)} - \partial_y 1/3 \rho^{(0)} u^{(1)}) + \rho^{(0)} u^{(1)} v^{(1)}. \tag{C.17}$$

The moment of second order $m_{220}^{*(2)}$ can be written as:

$$m_{220}^{*(2)} = \theta \rho^{(0)} m_{200}^{*(2)} + \theta \rho^{(0)} m_{020}^{*(2)} - \theta^2 \rho^{(2)}, \tag{C.18}$$

where:

$$\begin{aligned} m_{200}^{*(2)} &= (1-\omega_1) m_{200}^{(2)} + \omega_1 m_{200}^{eq(2)}, \\ m_{020}^{*(2)} &= (1-\omega_1) m_{020}^{(2)} + \omega_1 m_{020}^{eq(2)}, \end{aligned} \tag{C.19}$$

and [43]:

$$\begin{aligned} m_{200}^{(2)} &= m_{200}^{*(2)} - (\partial_x m_{100}^{eq(1)} + \partial_y m_{210}^{eq(1)} + \partial_z m_{201}^{eq(1)}), \\ m_{020}^{(2)} &= m_{020}^{*(2)} - (\partial_x m_{120}^{eq(1)} + \partial_y m_{010}^{eq(1)} + \partial_z m_{021}^{eq(1)}), \end{aligned} \tag{C.20}$$

The following terms are introduced:

$$\begin{aligned} m_{200}^{eq(2)} &= \theta \rho^{(2)} + {u^{(1)}}^2 \rho^{(0)}, \\ m_{020}^{eq(2)} &= \theta \rho^{(2)} + {v^{(1)}}^2 \rho^{(0)}, \\ m_{100}^{eq(1)} &= \rho^{(0)} u^{(1)}, \\ m_{201}^{eq(1)} &= m_{021}^{eq(1)} = \theta \rho^{(0)} w^{(1)} = 1/3 \rho^{(0)} w^{(1)}, \\ m_{010}^{eq(1)} &= \rho^{(0)} v^{(1)}. \end{aligned} \tag{C.21}$$

By inserting Eqs. (C.21) into Eqs. (C.20), and Eqs. (C.20) into Eq. (C.19), it is possible to write:

$$\begin{aligned} m_{200}^{*(2)} &= \theta \rho^{(2)} + {u^{(1)}}^2 \rho^{(0)} - \frac{(1-\omega_1)}{\omega_1}(\partial_x \rho^{(0)} u^{(1)} + \partial_y 1/3 \rho^{(0)} v^{(1)} + \partial_z 1/3 \rho^{(0)} w^{(1)}), \\ m_{020}^{*(2)} &= \theta \rho^{(2)} + {v^{(1)}}^2 \rho^{(0)} - \frac{(1-\omega_1)}{\omega_1}(\partial_x 1/3 \rho^{(0)} u^{(1)} + \partial_y \rho^{(0)} v^{(1)} + \partial_z 1/3 \rho^{(0)} w^{(1)}). \end{aligned} \tag{C.22}$$

By inserting Eqs. (C.22) into Eq. (C.18), the moment $m_{220}^{*(2)}$ becomes:

$$\begin{aligned} m_{220}^{*(2)} = {}& 1/3 \left(1/3 \rho^{(2)} + u^{(1)^2} \rho^{(0)} - \frac{(1-\omega_1)}{\omega_1} (\partial_x \rho^{(0)} u^{(1)} + \partial_y 1/3 \rho^{(0)} v^{(1)} + \partial_z 1/3 \rho^{(0)} w^{(1)}) \right) \\ & + 1/3 \left(1/3 \rho^{(2)} + v^{(1)^2} \rho^{(0)} - \frac{(1-\omega_1)}{\omega_1} (\partial_x 1/3 \rho^{(0)} u^{(1)} + \partial_y \rho^{(0)} v^{(1)} + \partial_z 1/3 \rho^{(0)} w^{(1)}) \right) \\ & - 1/9 \rho^{(2)}. \end{aligned} \tag{C.23}$$

The moment of second order $m_{211}^{*(2)}$ can be written as:

$$m_{121}^{*(2)} = \theta m_{101}^{*(2)}. \tag{C.24}$$

The moment $m_{101}^{*(2)}$ can be derived like the moment $m_{110}^{*(2)}$ in Eqs. (C.14)–(C.17). It reads:

$$m_{101}^{*(2)} = \frac{(1-\omega_1)}{\omega_1} (-\partial_x 1/3 \rho^{(0)} w^{(1)} - \partial_z 1/3 \rho^{(0)} u^{(1)}) + \omega_1 \rho^{(0)} u^{(1)} w^{(1)}. \tag{C.25}$$

The last step is to insert the Navier-Stokes momentum equation into the term $\partial_t m_{120}^{(1)}$:

$$\partial_t m_{120}^{(1)} = \partial_t \theta m_{100}^{(1)} = \partial_t 1/3 \rho^{(0)} u^{(1)}. \tag{C.26}$$

The Navier-Stokes momentum equation for the X-component reads:

$$\begin{aligned} \partial_t u^{(1)} = {}& -\partial_x u^{(1)^2} - \partial_y u^{(1)} v^{(1)} - \partial_z u^{(1)} w^{(1)} - \frac{1}{\rho^{(0)}} \partial_x p \\ & + \frac{1}{3} \left(\frac{1}{\omega_1} - \frac{1}{2} \right) \left[\partial_{xx} u^{(1)} + \partial_{yy} u^{(1)} + \partial_{zz} u^{(1)} \right] + g_x. \end{aligned} \tag{C.27}$$

Finally it is possible to rewrite Eq. (C.8) as:

$$\begin{aligned} m_{120}^{*(3)} - m_{120}^{(3)} = {}& 1/3 \rho^{(0)} \partial_t u^{(1)} + \partial_x m_{220}^{*(2)} + \partial_y m_{110}^{*(2)} + \partial_z m_{121}^{*(2)} \\ & - 1/2 (\partial_{xx} m_{120}^{*(1)} + \partial_{yy} m_{120}^{*(1)} + \partial_{zz} m_{122}^{*(1)}) \\ & - \partial_{xy} m_{210}^{*(1)} - \partial_{xy} m_{221}^{*(1)} - \partial_{yz} m_{111}^{*(1)}. \end{aligned} \tag{C.28}$$

The last step is to insert Eqs. (C.27), (C.25), (C.23), (C.17), (C.13), (C.12), (C.11), (C.10), and (C.9) into Eq. (C.28). By changing the moments with the cumulants, it is possible to obtain the relationship of Eq. (5.6):

$$C_{120}^{*} - C_{120} = -\frac{2}{9} \rho \left(\frac{1}{\omega_1} - \frac{1}{2} \right) [2 \partial_{xy} v + \partial_{yy} u] + 1/3 \rho g_x + \mathcal{O}(\Delta x^2). \tag{C.29}$$

Bibliography

[1] Peter A. Davidson, Yukio Kaneda, Keith Moffatt, and Katepalli R. Sreenivasan. *A voyage through turbulence*. Cambridge University Press, 2011.

[2] M. Islam, F. Decker, E. de Villiers, A. Jackson, J. Gines, T. Grahs, A. Gitt-Gehrke, and J. Comas i Font. Application of detached-eddy simulation for automotive aerodynamics development. *SAE Pap.*, pages 2009–01–0333, 2009.

[3] F. Durst. *Fluid Mechanics - An Introduction to the Theory of Fluid Flows*. Springer, 2008.

[4] G. Hauke. *An Introduction to Fluid Mechanics and Transport Phenomena*. Springer, 2008.

[5] NASA. Navier-Stokes Equations. . URL https://www.grc.nasa.gov/www/k-12/airplane/nseqs.html.

[6] NASA. Similarity Parameters. . URL https://www.grc.nasa.gov/www/k-12/airplane/airsim.html.

[7] H. v. Helmholtz. *Zwei Hydrodynamische Abhandlungen*. Leipzig Verlag Von Wilhelm Engelmann, 1896.

[8] Michael Leschziner. *Statistical Turbulence Modelling for Fluid Dynamics - Demystified: An introductory text for graduate Engineering students*. Imperial College Press, 2016.

[9] Haecheon Choi and Parviz Moin. Grid-point requirements for large eddy simulation: Chapman's estimates revisited. *Phys. Fluids*, 24(1):1–6, 2012.

[10] O. Reynolds. An experimental investigation of the circumstances which determine whether the motion of water shall be direct or sinuous and the law of resistance in parallel channels. *Phil. Trans. Roy. Soc.*, 174(1):935–982, 1883.

[11] H. K. Versteeg and W Malalasekera. *An introduction to Computational Fluid Dynamics: The Finite Volume Method*. Longman Scientific & Technical, 1995.

[12] Peter Bradshaw. Turbulent Secondary Flows. *Annu. Rev. Fluid Mech.*, 19(1): 53–74, 1987.

[13] F. R. Menter. Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA J.*, 32(8):1598–1605, 1994.

[14] F. R. Menter and Y. Egorov. The scale-adaptive simulation method for unsteady turbulent flow predictions. part 1: Theory and model description. *Flow, Turbul. Combust.*, 85(1):113–138, 2010.

[15] Y. Egorov, F. R. Menter, R. Lechner, and D. Cokljat. The Scale-Adaptive Simulation Method for Unsteady Turbulent Flow Predictions . Part 2 : Application to Complex Flows. *Flow, Turbul. Combust.*, 85(1):139–165, 2010.

[16] S. B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[17] J. Fröhlich and W. Rodi. Introduction to Large-Eddy Simulation of turbulent flows. In *Clos. Strateg. Turbul. Transitional Flows*, chapter 8, pages 267–298. Cambridge University Press, 2002.

[18] J. Smagorinsky. General Circulation Experiments With the Primitive Equations. *Mon. Weather Rev.*, 91(3):99–164, 1963.

[19] Massimo Germano, Ugo Piomelli, Parviz Moin, and William H. Cabot. A dynamic subgrid-scale eddy viscosity model. *Phys. Fluids A Fluid Dyn.*, 3(7):1760, 1991.

[20] Kiyosi Horiuti. A new dynamic two-parameter mixed model for large-eddy simulation. *Phys. Fluids*, 9(11):3443, 1997.

[21] F. Ducros, F. Nicoud, and T. Poinsot. Wall-adapting local eddy-viscosity models for simulations in complex geometries. *Conf. Numer. Methods Fluid Dyn.*, 1998.

[22] F. Nicoud and F. Ducros. Subgrid-scale stress modelling based on the square of the velocity gradient tensor. *Flow, Turbul. Combust.*, 62(3):183–200, 1999.

[23] M. Weickert, G. Teike, O. Schmidt, and M. Sommerfeld. Investigation of the LES WALE turbulence model within the lattice Boltzmann framework. *Comput. Math. with Appl.*, 59(7):2200–2214, 2010.

[24] P. Quéméré and P. Sagaut. Zonal multi-domain RANS/LES simulations of turbulent flows. *Int. J. Numer. Methods Fluids*, 40(7):903–925, 2002.

[25] M. Breuer, B. Jaffrézic, and K. Arora. Hybrid LES-RANS technique based on a one-equation near-wall model. *Theor. Comput. Fluid Dyn.*, 22(3-4):157–187, 2008.

[26] Jochen Frohlich and Dominic von Terzi. Hybrid LES/RANS methods for the simulation of turbulent flows. *Prog. Aerosp. Sci.*, 44(5):349–377, 2008.

[27] Philippe R. Spalart. Detached-Eddy Simulation. *Annu. Rev. Fluid Mech.*, 41(1): 181–202, 2009.

[28] T. J. Craft. Wall Functions. *TPFE MSc Adv. Turbul. Model.*, 2011.

[29] H Tennekes and J L Lumley. *A First Course In Turbulence*. MIT Press, 1972.

[30] Sergio Hoyas and Javier Jiménez. Scaling of the velocity fluctuations in turbulent channels up to Reτ=2003. *Phys. Fluids*, 18(1):1–4, 2006.

[31] J. Bredberg. On the Wall Boundary Condition for Turbulence Models. Technical report, Chalmers University of Technology, 2000.

[32] B.E. Launder and D.B. Spalding. The numerical computation of turbulent flows. *Comput. Methods Appl. Mech. Eng.*, 3(2):269–289, 1974.

[33] C. C. Chieng and B. E. Launder. On the Calculation of Turbulent Heat Transport Downstream From an Abrupt Pipe Expansion. *Numer. Heat Transf.*, 3(2):189–207, 1980.

[34] R. W. Johnson and B. E. Launder. Discussion of "On the Calculation of Turbulent Heat Transport Downstream From an Abrupt Pipe Expansion". *Numer. Heat Transf.*, 5:493–496, 1982.

[35] A. J. Musker. Explicit Expression for the Smooth Wall Velocity Distribution in a Turbulent Boundary Layer. *AIAA J.*, 17(6):655–657, 1979.

[36] Peter A. Monkewitz, Kapil A. Chauhan, and Hassan M. Nagib. Self-consistent high-Reynolds-number asymptotics for zero-pressure-gradient turbulent boundary layers. *Phys. Fluids*, 19(11), 2007.

[37] Tsan-Hsing Shih, Louis A. Povinelli, Nan-Suey Liu, Mark G. Potapczuk, and J. L. Lumley. A Generalized Wall Function. *NASA*, 1999.

[38] Tsan-Hsing Shih, Louis A. Povinelli, Nan-Suey Liu, and Kuo-Huey Chen. Generalized Complex Wall Function Turbulent for Flows. 2000.

[39] T. J. Craft, S. E. Gant, H. Iacovides, and B. E. Launder. A new wall function strategy for complex turbulent flows. *Numer. Heat Transf. Part B*, 45:301–318, 2004.

[40] Johan Hoffman. Simulation of turbulent flow past bluff bodies on coarse meshes using general galerkin methods: Drag crisis and turbulent euler solutions. *Comput. Mech.*, 38(4-5):390–402, 2006.

[41] Niclas Jansson and Johan Hoffman. Computer simulation of incompressible flow past a circular cylinder at very high Reynolds numbers. 2011.

[42] Ulrich Rüde, Karen Willcox, Lois Curfman McInnes, and Hans De Sterck. Future Directions in CSE Education and Research. Technical report, Workshop Sponsored by the Society for Industrial and Applied Mathematics (SIAM) and the European Exascale Software Initiative (EESI-2), 2015.

[43] Martin Geier, Martin Schönherr, Andrea Pasquali, and Manfred Krafczyk. The cumulant lattice Boltzmann equation in three dimensions: Theory and validation. *Comput. Math. with Appl.*, 70(4):507–547, 2015.

[44] Next Limit Thecnologies. XFlow 2016 COMPUTATIONAL FLUID DYNAMICS Product Sheet. 2016. URL http://xflowcfd.com/.

[45] Palabos. Lattice Boltzmann Method - the kernel of Palabos. URL http://www.palabos.org/software/lattice-boltzmann-method.

[46] I. Ginzbourg and P. M. Adler. Boundary flow condition analysis for the 3-Dimensional lattice Boltzmann model. *J. Phys. II*, 4(2):191–214, 1994.

[47] Irina Ginzburg and Dominique D'Humières. Multireflection boundary conditions for lattice Boltzmann models. *Phys. Rev. E. Stat. Nonlin. Soft Matter Phys.*, 68 (6 Pt 2):066614, 2003.

[48] O. Malaspinas and P. Sagaut. Wall model for large-eddy simulation based on the lattice Boltzmann method. *J. Comput. Phys.*, 275:25–40, 2014.

[49] Beryl Lieff Benderly. How scientific culture discourages new ideas. *Sci. AAAS*, 2016.

[50] David P. Lockard, Li Shi Luo, Seth D. Milder, and Bart A. Singer. Evaluation of PowerFLOW for aerodynamic applications. *J. Stat. Phys.*, 107(1-2):423–478, 2002.

[51] David M Holman, Ruddy M Brionnaud, and Zaki Abiza. Solution to industry benchmarck problems with the Lattice-Boltzmann code XFlow. *Eur. Congr. Comput. Methods Appl. Sci. Eng. (ECCOMAS 2012)*, page 22, 2012.

[52] D. D'Humières. Generalized lattice Boltzmann equations. In Rarefied gas dynamics: theory and simulations. *Prog. Aeronaut. Astronaut.*, 159:450–458, 1992.

[53] Ludwig Boltzmann. Further Studies on the Thermal Equilibrium of Gas Molecules. *Kinet. Theory*, 2:88–175, 1872.

[54] Ludwig Boltzmann. Vorlesungen über Gastheorie. *Barth, Leipzig*, 1, 1896.

[55] P. L. Bhatnagar, E. P. Gross, and M. Krook. A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems. *Phys. Rev.*, 94(3):511–525, 1954.

[56] R. Benzi, S. Succi, and M. Vergassola. The lattice Boltzmann equation: theory and applications. *Phys. Rep.*, 222(3):145–197, 1992.

[57] S. Succi. *The lattice Boltzmann equation for Fluid Dynamics and Beyond*. Oxford University press, 2001.

[58] Y H Qian, D D'Humières, and P Lallemand. Lattice BGK Models for Navier-Stokes Equation. *EPL (Europhysics Lett.*, 17:479–484, 1992.

[59] Dominique D'Humières, Irina Ginzburg, Manfred Krafczyk, Pierre Lallemand, and Li-Shi Luo. Multiple-relaxation-time lattice Boltzmann models in three dimensions. *Philos. Trans. A. Math. Phys. Eng. Sci.*, 360(1792):437–451, 2002.

[60] Martin Christian Geier. *Ab initio derivation of the cascaded lattice boltzmann automaton*. Ph.d. thesis, University of Freiburg, 2006.

[61] Martin Geier, Andreas Greiner, and Jan G. Korvink. Cascaded digital lattice Boltzmann automata for high Reynolds number flow. *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, 73(6):1–10, 2006.

[62] M. Geier, A. Greiner, and J. G. Korvink. A factorized central moment lattice Boltzmann method. *Eur. Phys. J. Spec. Top.*, 171(1):55–61, 2009.

[63] H. Clarke, D. and Salas, M. and Hassan. Euler Calculations for Multi-Element Airfoils Using Cartesian Grids. *AIAA J.*, 24(3):1128–1135, 1986.

[64] M'hamed Bouzidi, Mouaouia Firdaouss, and Pierre Lallemand. Momentum transfer of a Boltzmann-lattice fluid with boundaries. *Phys. Fluids*, 13(11):3452–3459, 2001.

[65] M. Schönherr, K. Kucher, M. Geier, M. Stiebler, S. Freudiger, and M. Krafczyk. Multi-thread implementations of the lattice Boltzmann method on non-uniform grids for CPUs and GPUs. *Comput. Math. with Appl.*, 61(12):3730–3743, 2011.

[66] Ehsan Goraki, Martin Geier, Kostyantyn Kucher, and Manfred Krafczyk. Distributed cumulant lattice Boltzmann simulation of the dispersion process of ceramic agglomerates. *Jounal Comput. methods Sci. Eng.*, 16(2):231–252, 2016.

[67] Andrea Pasquali, Martin Schönherr, Martin Geier, and Manfred Krafczyk. Simulation of external aerodynamics of the DrivAer model with the LBM on GPGPUs. *Adv. Parallel Comput.*, 27:391 – 400, 2016.

[68] Martin Schönherr. *Towards reliable LES-CFD computations based on advanced LBM models utilizing (Multi-) GPGPU hardware.* Ph.d. thesis, Technische Universität Braunschweig, 2015.

[69] Jan Linxweiler. *Ein integrierter Softwareansatz zur interaktiven Exploration und Steuerung von Strömungssimulationen auf Many-Core-Architekturen.* Ph.d. thesis, Technische Universität Braunschweig, 2011.

[70] M. Geier, A. Greiner, and J. G. Korvink. Bubble functions for the lattice Boltzmann method and their application to grid refinement. *Eur. Phys. J. Spec. Top.*, 171(1):173–179, 2009.

[71] Andrea Pasquali, Martin Geier, and Manfred Krafczyk. LBMHexMesh: an OpenFOAM based grid generator for the Lattice Boltzmann Method (LBM). *7th Open Source CFD Int. Conf.*, 2013.

[72] OpenFOAM Foundation. OpenFOAM The Open Source CFD Toolbox User Guide Version 2.2.0. (August), 2013. URL http://openfoam.org/.

[73] Marshall Burns. StereoLithography Interface Specification. *3D Syst. Inc.*, 1988.

[74] Tomas Möller and Ben Trumbore. Fast, Minimum Storage Ray/Triangle Intersection.

[75] Edgar Gabriel, Graham E Fagg, George Bosilca, Thara Angskun, Jack J Dongarra, Jeffrey M Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett,

Andrew Lumsdaine, Ralph H Castain, David J Daniel, Richard L Graham, and Timothy S Woodall. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. *11th Eur. PVM/MPI Users' Gr. Meet.*, 2004.

[76] William J Schroeder, Kenneth M Martin, and William E Lorensen. The Design and Implementation Of An Object-Oriented Toolkit For 3D Graphics And Visualization. *IEEE Vis.*, 1996.

[77] James Ahrens, Berk Geveci, and Charles Law. ParaView: An End-User Tool for Large Data Visualization. *Vis. Handb.*, 2005.

[78] Bundesanstalt für Wasserbau. Principles for the Design of Bank and Bottom Protection for Inland Waterways (GBB). 2010.

[79] Angelina I Heft, Thomas Indinger, and Nikolaus A Adams. Introduction of a New Realistic Generic Car Model for Aerodynamic Investigations. *SAE Int.*, 2012.

[80] BETA CAE. ANSA META for multidisciplinary CAE pre- & pot-processing. URL http://www.beta-cae.com/.

[81] E. Achenbach. Experiments on the flow past spheres at very high Reynolds numbers. *Journal of Fluid Mechanics*, 54(2), 1972.

[82] L. Prandtl. Strmungsmechanik Bd. 5. *Universittsverlag Göttingen*, 2009.

[83] V. Bakic. *Experimental investigation of turbulent flows around a sphere.* Ph.d. thesis, Technische Universität Hamburg-Harburg, 2002.

[84] F. A. Morrison. *An Introduction to Fluid Mechanics.* Cambridge University Press, 2013.

[85] J. Almedeij. Drag coefficient of flow around a sphere: Matching asymptotically the wide trend. *Powder Technology*, 186(3):218–223, 2008.

[86] J.C.R. Hunt, A.A. Wray, and P. Moin. Eddies, streams, and convergence zones in turbulent flows. *Cent. Turbul. Res. Proc. Summer Progr.*, (1970):193–208, 1988.

[87] Lakshmi Narasiman Vijayasarathi. Simulation of an aeroacoustic flow in a pipe and experimental validation. Technical report, Technische Universität Braunschweig, 2016.

[88] Dassault Systèmes. CATIA. URL http://www.3ds.com/products-services/catia/.

[89] C. Kling. Private communications. 2016.

[90] Xiaofan Yang, Yashar Mehmani, William A. Perkins, Andrea Pasquali, Martin Schönherr, Kyungjoo Kim, Mauro Perego, Michael L. Parks, Nathaniel Trask, Matthew T. Balhoff, Marshall C. Richmond, Martin Geier, Manfred Krafczyk, Li Shi Luo, Alexandre M. Tartakovsky, and Timothy D. Scheibe. Intercomparison of 3D pore-scale flow and solute transport simulation methods. *Adv. Water Resour.*, (9), 2015.

[91] Marco Giometto. *Large eddy simulation of atmospheric boundary layer flow over a realistic urban surface*. Ph.d. thesis, Technische Universität Braunschweig, 2014.

[92] Stephan Lenz. Lattice-Boltzmann Simulation of a Turbulent Wind Flow Over a Real Urban Area. Technical report, Technische Universität Braunschweig, 2016.

[93] Stephan Lenz. Effiziente Simulation turbulenter Windströmungen im urbanen Raum. *28. Forum Bauinformatik*, 2016.

[94] George Karypis and Vipin Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.

[95] Hamburg University of Technology. elbe - an efficient lattice boltzmann environment. URL https://www.tuhh.de/elbe/home.html.

[96] Christian Janßen, Dennis Mierke, Micha Überrück, Silke Gralher, and Thomas Rung. Validation of the GPU-Accelerated CFD Solver ELBE for Free Surface Flow Problems in Civil and Environmental Engineering. *Computation*, 3(3):354–385, 2015.

[97] Nils Koliha. *Computation and Real-Time Rendering of Complex Multiphase Flows on GPGPU Clusters*. Master thesis, Hamburg University of Technology, 2013.

[98] C. F. Janßen, N. Koliha, and T. Rung. A fast and rigorously parallel surface voxelization technique for GPU-accelerated CFD simulations. *Commun. Comput. Phys.*, 17(5):1246–1270, 2015.

[99] S.M.J. Guzik, Xinfeng Gao, Todd Weisgraber, Berni Alder, and Phillip Colella. An Adaptive Mesh Refinement Strategy with Conservative Space-Time Coupling for the Lattice-Boltzmann Method. *Am. Inst. Aeronaut. Astronaut.*, pages 1–11, 2014.

[100] Abbas Fakhari and Taehun Lee. Finite-difference lattice Boltzmann method with a block-structured adaptive-mesh-refinement technique. *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, 89(3):1–12, 2014.

[101] Dominique D'Humières and Irina Ginzburg. Viscosity independent numerical errors for Lattice Boltzmann models: From recurrence equations to "magic" collision numbers. *Comput. Math. with Appl.*, 58(5):823–840, 2009.

[102] François Dubois, Pierre Lallemand, and Mahdi Tekitek. On a superconvergent lattice Boltzmann boundary scheme. *Comput. Math. with Appl.*, 59(7):2141–2149, 2010.

[103] M. Geier. Private communications. 2016.

[104] Martin Geier, Martin Schönherr, Andrea Pasquali, and Manfred Krafczyk. Simulating the drag crisis behind a sphere with the lattice Boltzmann method: do or die. *13th Int. Conf. Mesoscopic Methods Eng. Sci.*, 2016.

[105] François Dubois. Equivalent partial differential equations of a lattice Boltzmann scheme. *Comput. Math. with Appl.*, 55(7):1441–1449, 2008.

[106] Michael Junk, Axel Klar, and Li-Shi Luo. Asymptotic analysis of the lattice Boltzmann equation. *J. Comput. Phys.*, 210(2):676–704, 2005.

[107] G. I. Taylor and A. E. Green. Mechanism of the Production of Small Eddies from Large Ones. *Proc. R. Soc. A Math. Phys. Eng. Sci.*, 158(895):499–521, 1937.

[108] Johanna Vogel-Prandtl. *Ludwig Prandtl. A Personal Biography Drawn from Memories and Correspondence (Translated by D. A. Tigwell).* Universitätsverlag Göttingen, 2014.

[109] Herman Schlichting. *Boundary-Layer Theory (Translated by J. Kestin).* McGraw-Hill Book Company, 1979.

[110] John D. Anderson Jr. Ludwig Prandtl's Boundary Layer. *Phys. Today*, (12), 2005.

[111] K. T. Trinh. On the Blasius correlation for friction factors. 2010.

[112] D. R. Chapman. Computational aerodynamics development and outlook. *AIAA J.*, 17(12):1293–1313, 1979.

[113] B.S. Baldwin and H. Lomax. Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows. *AIAA 16th Aerosp. Sci. Meet.*, page 9, 1978.

[114] T. J. Craft. Numerical Solution of Boundary Layer Equations Example. *3rd Year Fluid Mech.*, 2008.

[115] S. Bocquet, P. Sagaut, and J. Jouhaud. A compressible wall model for large-eddy simulation with application to prediction of aerothermal quantities. *Phys. Fluids*, 24(6), 2012.

[116] Christophe Ybert, Catherine Barentin, Cécile Cottin-Bizonne, Pierre Joseph, and Lydéric Bocquet. Achieving large slip with superhydrophobic surfaces: Scaling laws for generic geometries. *Phys. Fluids*, 19(12):17–19, 2007.

[117] Olivier Cabrit and Franck Nicoud. Direct simulations for wall modeling of multicomponent reacting compressible turbulent flows. *Phys. Fluids*, 21(5), 2009.

[118] Manjusha Namburi, Siddharth Krithivasan, and Santosh Ansumali. Crystallographic Lattice Boltzmann Method. *Sci. Rep.*, 6(January):27172, 2016.

[119] Accademia del Cimento. *Atti e memorie inedite dell'Accademia del Cimento e notizie aneddote dei progressi delle scienze in Toscana contenenti memorie, esperienze, osservazioni, scoperte e la rinnovazione della fisica celeste e terrestre, cominciando da Galileo Galilei*. In Firenze : si vende da Giuseppe Tofani stampatore e da Luigi Carlieri librajo, 1780.

[120] University of Waterloo Libraries and Scholarly Societies Project. Accademia del Cimento. URL http://www.scholarly-societies.org/history/1657ac.html.